# A multi-sensor traffic scene dataset with omnidirectional video

Philipp Koschorrek[1], Tommaso Piccini[1], Per Öberg[2], Michael Felsberg[1], Lars Nielsen[2], Rudolf Mester[1,3]
[1]Computer Vision Laboratory, Dept. of Electrical Engineering, Linköping University, Sweden
[2]Vehicular Systems, Dept. of Electrical Engineering, Linköping University, Sweden
[3]VSI Lab, Computer Science Dept., Univ. Frankfurt, Germany
{firstname}.{lastname}@liu.se

## Abstract

*The development of vehicles that perceive their environment, in particular those using computer vision, indispensably requires large databases of sensor recordings obtained from real cars driven in realistic traffic situations. These datasets should be time shaped for enabling synchronization of sensor data from different sources. Furthermore, full surround environment perception requires high frame rates of synchronized omnidirectional video data to prevent information loss at any speeds.*

*This paper describes an experimental setup and software environment for recording such synchronized multi-sensor data streams and storing them in a new open source format. The dataset consists of sequences recorded in various environments from a car equipped with an omnidirectional multi-camera, height sensors, an IMU, a velocity sensor, and a GPS. The software environment for reading these data sets will be provided to the public, together with a collection of long multi-sensor and multi-camera data streams stored in the developed format.*

## 1. Introduction

Computer vision is evolving into a key technology for the automotive area, starting from simple, mostly visualization related tasks such as rear cameras, to driver assistance functionalities like lane-keeping assistants, traffic sign recognition, and soon including the field of visually guided autonomous driving. It is obvious already today that multiple cameras will be present in future cars, thus allowing to monitor the complete 360° field of view of a car in flowing traffic, detect dangerous situations and increase safety and comfort in driving. These prospects depend crucially on the availability of intelligent algorithms, evaluated and optimized on data of real traffic situations.

Building a dynamic environment model from a stream of data coming from a set of cameras looking into multiple directions around a vehicle is a challenging problem. Further challenges arise from the immense variability of scenes under different weather and illumination situations, considering occlusions and sensor imperfections, that make images difficult or ambiguous to be interpreted.

Figure 1. Test platform car with roof rack mounted Ladybug 3

In order to develop robust and reliable algorithms for camera based environment sensing and interpretation in multi-view scenarios, large realistic datasets are needed.

Our contributions are a framework for storing synchronized multi-sensor multi-camera data and example datasets, suitable for testing solutions to tasks such as visual odometry, SLAM, sensor fusion, motion estimation and optical flow computation.

Such datasets provide several significant benefits:

- It is possible to test algorithms from a real experimental car without investing time and money to actually build such a setup and record sequences with it.
- Evaluation on a standard dataset allows direct comparison of algorithms and methods.
- Thanks to redundant sensors, ground truth for e.g. egomotion can be determined by sensor fusion.

Data that is of primary interest besides video is a) the true ego-motion of the car w.r.t. the static environment, and b) the true 3D structure of the environment, including the static scene and other moving objects.

### 1.1. Motivation

Our aim is to develop algorithms working in real urban or highway traffic situations, including high speed ranges. Hence, high frame rates and real traffic sequences are needed. Since none of the existing datasets provides such information in a way which also allows real time simulations, we decided to provide a new dataset, aimed mainly at visual odometry applications. The dataset features the kind of information we consider to be lacking in previous work, and a flexibility that allows to represent any kind of multisensor setups in the same framework. The experimental setup we used includes an omnidirectional camera, an IMU, laser-based height sensors, a velocity sensor based on optical road surface correlation, and a GPS receiver, mounted on a roadworthy car allowing us to take part in real traffic situations. All recorded data are stored in a new, flexible format using common timestamps, which makes real time playbacks possible. We provide also the necessary software environment and APIs to access data for different popular programming languages. Existing algorithms for visual odometry with a similar setup like the one proposed of Tardif et al. [7] can be revisited and compared against a common ground truth, even when driving in regions where absolute reference (e.g. GPS) is not available.

### 1.2. Ground Truth

While the 6D pose can be extracted from auxiliary sensors, by design, we chose not to include such information in our dataset. This choice is dictated by our view on what should be considered ground truth. In the case of real life scenarios we do not have access to the true model parameters (as we have for synthetic data) and we have to rely on measurements and sensor data from independent sensors. Moreover those may be afflicted by accidental or systematic distortions in the measuring procedure or in the sensor readings. At best, these distortions are statistically independent of the measurements under assessment. Also since the state-of-the-art in fusion methods continuously evolves, storing raw data instead of processed data ensures sustainability. We therefore prefer to leave the choice on the algorithm to determine 'ground truth' to each final user of our data, while groups interested in sensor fusion may gain interest in our data and contribute to a, possibly evolving, gold standard for ground truth in our sequences. Another factor that brought us to our choice is that it is generally preferable if ground truth is extracted from the data by an independent third party as suboptimal ground truth can constitute a strong bias in algorithm evaluation. For all these reasons, we decided to store the raw sensor recordings.

While we are using mid-level IMU and GPS sensors, they can be fused with further high accuracy sensor data (height and speed sensors, 2.1) to obtain ground truth.

Similarly, the available data can be used for evaluating Structure-From-Motion (SFM) algorithms. While we do not provide any ground truth for the 3D structure of the scenarios, a highly accurate model can be obtained from state-of-the-art bundle adjustment on all data of a sequence together with the provided calibration parameters. Such a model may be useful for evaluations of SFM algorithms on subsets of data of the corresponding sequence. A similar approach is used in Middlebury data [5].

### 1.3. Related work

Automotive, vision related datasets have been published by different research groups in the past few years.

The KITTI dataset and benchmark suite [3] provides combined IMU and GPS data, point clouds produced by a laser scanner and stereo visual data in both grayscale and color format. The primary aim of the KITTI dataset is, however, to provide a benchmark suite for the testing of computer vision algorithms on real automotive applications. It is thus composed of several short sequences consisting in challenges of levels of difficulty for algorithms. Though this is an ideal approach for testing and benchmarking of algorithms, when the aim is the design of robust, industry oriented software, much longer and diverse sequences showing as various situations as possible are needed. For comparison, the size of the whole KITTI dataset is of approximately 20GB, about equivalent to the size of our shortest sequence after compression.

Blanco et al. present in [1] a dataset with a focus on "centimeter-accuracy ground truth" of position data for testing SLAM algorithms designed for small mobile robots. Their dataset is composed of synchronized data coming from multiple sensors (color cameras, laser scanners, different kinds of GPS receivers and IMU sensors) and provided in a structured, plain text file based format together with an API to handle the data. The data is, however, taken at slow speeds from a steady and controlled environment such as a university campus and a parking lot. In a later extension of their work, data has been gathered also for a longer sequence taken in a urban scenario[1]. Their setup misses, however, the omnidirectional visual data that we consider of fundamental importance for full surround sensing.

We identified only two publicly available datasets that show significant affinity to our work. One is the work presented by Smith et al. [6]. Visual and inertial data is gathered with a multisensor setup based on a Segway like vehicle driven through the New College Campus in Oxford, UK. The dataset comprises odometry information, GPS data and point clouds provided with a couple of laser scanners together with visual data coming from both a stereo pair of grayscale cameras at 20 frames per second (fps) and an omnidirectional color camera (3 fps). The dataset is provided in a structured format where the sensor readings are in a

[1] http://www.mrpt.org/MalagaUrbanDataset

time stamped, time ordered text file, separated from the visual data. This approach is very similar to our own, but the plain text format used introduces a lot of unnecessary redundancy in the data which also results in a increased size of the files while giving in exchange only a small advance in data handling. Pandey at al. [4] published a second dataset similar to our own. They provide recordings of omnidirectional visual data at 8 fps together with laser scans and inertial sensor readings. The data presented have, however, to be synchronized in post-processing since the readings of the sensors are stored separately. Furthermore the dataset consists only of two sequences.
While the merits of these two works are undeniable, they both still fall short in aspects that we consider of primary importance, such as the frame rate of the omnidirectional visual data, the environment chosen for the data recordings and the possibility for real-time simulation. The omnidirectional visual data is provided at only 3 and 8 fps respectively, and in both cases the sequences were recorded inside the respective university campuses where no other or only few vehicles are present. The vehicle speed and the variability of the surrounding landscape within the sequences, as well as between the sequences, is low compared to our dataset. Furthermore their APIs do not provide a comfortable method for the streaming of the data.

## 2. Experimental setup

In this section, we provide a detailed description of our experimental setup in both hardware and software aspects.

### 2.1. Hardware

The dataset was recorded with a Volkswagen Golf 5 (Fig. 1) equipped with an omnidirectional multi-camera system, a Point Grey Ladybug 3[2] (C), and several sensors recording the motion of the car both in the car's own frame of reference and a global coordinate system. The camera consists of 6 synchronized global shutter color cameras configured to a raw resolution of $616 \times 1616$ pixel ($width \times height$) each, stored as Bayer images. While one camera is pointing upwards and will be of marginal interest for most applications, the five remaining cameras depict the horizontal surroundings of the car. The arrangement of the cameras with overlapping fields of view allows a $360^\circ$ view of the environment. The frame rate of 30 fps is much higher than in similar datasets [6, 4].
Such a setup is chosen for the flexibility it provides. While an omnidirectional camera positioned on the roof of a car will hardly be the choice of a car producer, experimentally it allows to simulate any case of camera setting with slightly overlapping fields of view. Omnidirectional stereo vision is theoretically possible using two LadyBug cameras stacked on top of each other, but this is practically unfeasible due to mechanical and data-rate constraints. Localized stereo vision, on the other hand, could be easily introduced in the future, if deemed necessary, by e.g. adding a single forward looking camera.

Several other sensors complement the camera in order to obtain an extended and redundant set of measurements from which egomotion and dynamic attitude can be computed. The sensors are the following:

- camera mounted IMU
- lateral & longitudinal velocity sensor (car front)
- three non contact laser ride height sensors (side doors and car front)
- GPS receiver

Figure 2 shows the mechanical setup, whereas the sensor setup of the car is shown in Figure 3, with the multi-camera rig being shifted in the drawing from the roof. A more detailed sketch of the car setup can be found on the project website (see Section 4).

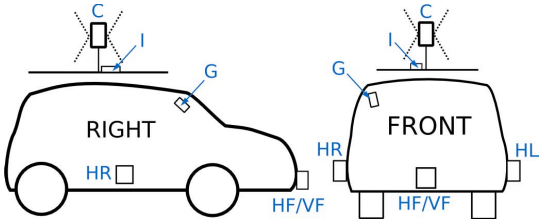

Figure 2. Mechanical setup: C - Camera, I - IMU, G - GPS receiver, HR, HL, HF - Height sensors, VF - Velocity sensor

The XSens MTi AHRS[3] (I) is an IMU with 9 channels which measures the acceleration, angular velocity and magnetic field in all three directions of its internal coordinate system. The velocity sensor (VF) and the height sensors mounted at the front (HF) and at the side doors (HR, HL, right and left respectively) of the car provide additional information such as more accurate estimates for the attitude and speed of the car to optimize and stabilize state estimations based on IMU data. Additionally to the longitudinal velocity VF measures the lateral velocity, too. The height sensors are mounted on the front and the sides of the car allowing an estimation of the roll and pitch angle of the car. The velocity sensor, a Kistler Correvit S-350 Aqua, and the height sensors, Kistler HF-500C, are parts of the *3-axis optical measurement system* by Corrsys Datron[4]. The GPS receiver is a uBlox AEK-4P GPS development kit using only pure GPS information without RTK correction signals.
The Ladybug camera has been setup in such a way that one camera points in the forward direction of the car, while the viewing directions of the four remaining cameras are to the

[2]http://www.ptgrey.com/products/ladybug3/

[3]http://www.xsens.com/en/general/mti

[4]http://www.corrsys-datron.com/optical_sensors.htm

side and rear of the car with an angle of 72° between adjacent cameras. Due to the distances between the sensors, each sensor has to be described in its own coordinate system. These coordinate systems are named after the sensors plus a suffix e.g. HRCS, HLCS, etc. All sensor coordinate systems are right-handed and set in reference to the coordinate system of the car (CARCS) which lies in the geometric center of the bounding box of the car chassis, i.e., the car without tires and suspension. CARCS is also a right-handed coordinate system where the x-axis is pointing forward and the y-axis to the right of the car whereas the axes are parallel to the bounding box edges of the car. The attitude of the car describes the rotation of this coordinate system to a coordinate system SCS which lies on the street with x pointing forward and z parallel to the downwards directed surface normal, i.e. to the model of a plain street. The relations of the sensor coordinate systems to the car reference frame as well as the calibration data of the camera done with the framework provided by [2] can be found on the project web page, see Section 4. Furthermore, the known calibration parameters allow us to merge the six camera recordings according to different camera models, e.g. panorama images, dome projections or spherical projections. The sampling frequencies $F_{Si}$ of the different sensors are shown in Table 1.

Table 1. Sampling frequencies

| Sensor | $F_{Si}$[Hz] |
|---|---|
| C | 30 |
| I | 100 |
| VF | 250 |
| HR, HL, HF | 250 |
| GPS | $\sim 4$ |

Furthermore, the car internal CAN (Controller Area Network) bus messages are recorded in raw format, too, but are not yet further processed.
The measurements gathered by the IMU and GPS have not been combined. The absolute position and velocity of the car in the world reference frame obtained by the GPS module and the measured IMU data in its coordinate system are stored separately. Attitude estimates for the IMU and, due to their rigid coupling, also for the camera are produced by the internal Kalman Filter of the IMU and stored, as well.
Obviously a fusion of all sensor data or at least a few, e.g. IMU and GPS, would improve the accuracy of the estimated position, but it has not been applied due to the reasons already mentioned in Section 1.2.
The sensors acquire their measurements continuously at the mentioned sampling frequencies and send them to a computer, which is referred to as *LinCom* in the following.

The camera is connected via FireWire to and controlled from a second computer, *WinCom*, controlling only the multi-camera due to high necessary data rates. LinCom is controlling and reading the remaining sensors to obtain measurements in real-time. Both computers are connected via Ethernet in a Local Area Network in order to provide time synchronization.

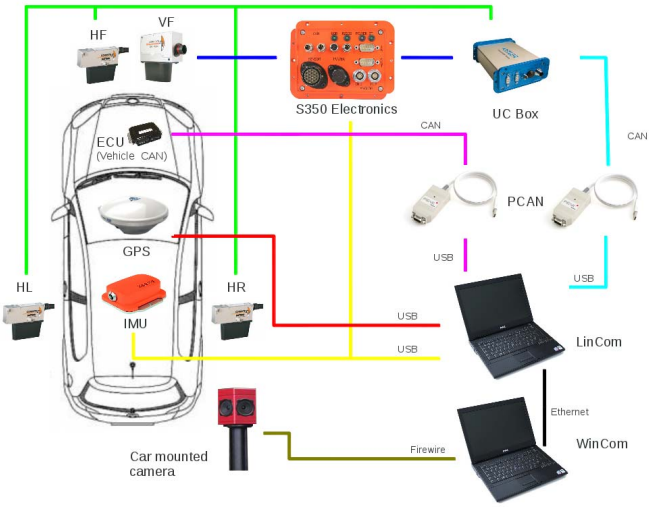


Figure 3. Sensor arrangement scheme

## 2.2. Software

This section describes the software running on the different computers as well as the method to synchronize them.

**Data synchronization** For the combination of sensor data and images, it is necessary that they share the same time base. In consequence of the need of two computers, their internal clocks have to be synchronized to achieve this goal. In order to provide high accuracy, a synchronization via Network Time Protocol (NTP) was chosen. LinCom was adapted to act as a NTP server sending out time information. The *stratum* value of the NTP server was set very low to make sure that the connected computer, WinCom, trusts the time signal. Obviously it is not necessary to have an accurate global time, rather it is necessary for the computers to share the *same* time. LinCom sends out its own kernel time as time base and WinCom is synchronizing to this time. The time difference between the computers achieved with this method is, according to the NTP client, below 2ms.
Further time information such as global time (UTC) can be extracted from GPS recordings if this is required.

**WinCom** WinCom runs Windows 7 as operating system, as this is necessary for accessing the API of the Ladybug3 camera with the full feature set. A recording program was written in C/C++ to grab the images of the camera and store them. Besides the image handling, the software also records the time at which the image was grabbed. This time corresponds, except for the mentioned synchronization uncer-

tainty, to the kernel time of LinCom. With those time values and corresponding time stamps in the recordings of LinCom, a merging of the sensor data and image information is possible. Extra camera information, such as the shutter time, is also stored with the images.

**LinCom** A Gentoo Linux distribution without graphical user interface is running on LinCom to achieve maximal performances while having all benefits of a modern operating system. A real-time Linux could not been used since there are no real-time Linux USB stacks. The measurement program was written in C/C++ using the *Boost*[5] library and the APIs of the sensors and interface devices for CAN. It is thread-based for simultaneous event-based reading of all sensors. In consequence of the asynchronous transmission of the sensor data, it is necessary that all functions which collect measurements are running in parallel. Otherwise it cannot be avoided that sensor data capture is delayed, or even data is lost. The principal structure of the program has been adopted from [8] whereas now each sensor is assigned to its own thread, which waits in a loop for incoming data. After the reception of data, the current kernel time is determined and the difference of the current time and a reference time (the start of the measurement program) is used to calculate a time stamp of the incoming message which is stored together with the sensor data. After reading the raw data and calculating the time stamp of these messages, the readings are converted in physical values with a given unit of measure, e.g. $[\frac{m}{s^2}]$ for acceleration or $\frac{m}{s}$ for velocity. In the next step the time stamp, a unique identifier describing the sensor data and the converted data are written to a string and stored temporarily in a thread-coupled FIFO memory. Each sensor thread deals with the data like this. A round-robin scheduled thread periodically reads out the FIFOs one by one and stores the contained data strings in a CSV file.

## 3. Dataset structure

To the best of our knowledge, a standard file format for the storage of *large image sequences* with additional synchronized data such as IMU or GPS data has not been proposed in the public yet. There are already available, related storing methods such as the one used by ROS[6], *bag files*, which are capable of storing different kinds of data. We considered to store the data as bag files, but this has severe disadvantages, among which the fact that it is not stable on all platforms and only fully supported for Ubuntu Linux. Since ROS is needed to read the bag files, parts of the program have to be ROS nodes. Furthermore, the data would only be accessible through ROS *topics*. Hence a set of static ROS message definitions and publisher would be needed additionally to the bag file for describing the ROS message structure and publishing the topics. This makes usage of ROS less flexible here than our approach in terms of adding new sensors. The major problem though, is the size of the data. Due to the high data rates and the large number of cameras, the bag files would be monolithic files with several Gigabytes in size.
The format proposed here, the *LiU Stream Format* , instead, handles such combined data recordings under the constraints given by different file systems such as FAT32, NTFS, ext3 and ext4, on which a storage of the dataset should be possible. The images are stored as standard image files in a hierarchical but transparent folder structure to achieve a limited number of files per folder. The additional sensor data are stored in linked binary files, called *stream files*, organized as message blocks in temporal order. We propose this format as an Open Source format and provide APIs to read from it, as well as guidelines for expanding our work.

### 3.1. Stream File

In the stream file, each message corresponds to one measurement at a certain time and has a specific frame format (Figure 4).

| Messagestart | ID | Length of Payload | Payload ... |
|---|---|---|---|
| 4x *uint8* '$BST' | *uint32* | *uint32* | Dependent on message |

Figure 4. Format of a streamfile message

Each message consists of a header composed of three fixed parts: a block start signal ($BST), an identifier and a length variable. The block start is a unique character sequence to identify the begin of a message. It should not be too long to waste memory and not to short to avoid mismatches with random data. The identifier, the ID, describes the kind of message, i.e. the content of the message. Connected to a certain ID is a specific message structure as well as a specific sensor. Subsequent to the ID, the length of the message payload in bytes is stored. The actual message called payload follows; it is sensor- and user-dependent and can vary from sensor to sensor. Payloads have been specified for the sensors we used so far. As an example, Table 2 shows the payload format for IMU recordings. This payload is 80 byte long whereas the first 8 bytes are one *unsigned int* value and each of the following 8 bytes are *double* values.

All message payload structures, their corresponding IDs and additional information are described in a XML-file which is used to initialize the reading of the stream file, but serves also to describe the structure in a format accessible to human readers, and finally allows also ID and message pool expansion. This file is available together with the data set and the API; see more in Section 4.

[5] http://www.boost.org/
[6] http://www.ros.org/

Table 2. Structure of an IMU message payload with $\vec{a}$ as a three component vector $\vec{a} = \{a_x, a_y, a_z\}$

| Order | Description | Data type | Unit |
|---|---|---|---|
| 1 | Time stamp | *uint64* | $\mu s$ |
| 2 | Acceleration | 3x *double* $\vec{a}$ | $m/s^2$ |
| 3 | Angular velocity | 3x *double* $\vec{a}$ | $deg/s$ |
| 4 | Magnetic flux density | 3x *double* $\vec{a}$ | $Gs$ |

There are also message blocks for the beginning, the end and also the linking of stream file parts. The stream file is partitioned since the size of a stream file part has been limited. These meta data are not visible to the user; instead, by usage of the API, several stream files will appear as one.

### 3.2. File organization

The structure and organization of the proposed stream format is shown in Figure 5. Each stream of data, i.e. each

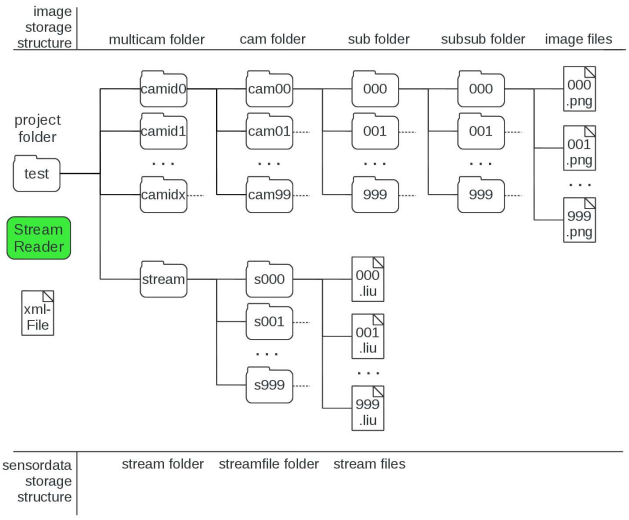


Figure 5. Hierarchical tree structure of the LiU stream format

*test drive* or *dataset*, corresponds to one *root folder* in the topmost level of a hierarchical file/folder structure. This root folder is named *test* in the example given by Fig. 5. In the next layer, the dataset consists of several folders, but at least two. There is always one folder containing the actual stream files, the *stream folder*, while the other folders are representing the multi-cameras: each folder one multi-camera. A multi-camera is a camera, which may have more than one optical sensor or camera, e.g. the Ladybug 3 appears as a multi-camera with six cameras. The name of those *multicam folders* are the unique identification numbers of the multi-cameras. Each single camera of a multi-camera is associated with its own folder *camxx* in the corresponding multi-camera folder where the $x$ are placeholders for the number of the camera with leading zeros. This folder contains all images recorded by this camera, each one stored as single image file in the Portable Networks Graphics (PNG) format. Two levels of sub-folders deeper in each cam folder, the images are stored. Each folder and image has a name consisting of a 3-digit number, beginning with *000* ending with *999*. With this naming convention it is possible to store up 1.000 elements in each folder. This choice is due both to avoid possible limitations imposed by file systems and to make browsing through the directory structure painless as loading a directory containing hundreds of thousands of images can be challenging even for modern systems. The composition of an image path of an image with a given number is easily done with this structure without providing more information than the image number and the multi-camera which captured the image. This makes an image-only reading possible.

It is important to mention that there has to be the same number of images for every camera. That would mean that every image with the image number 0 would be an image recorded at a certain time by all cameras of a corresponding multi-camera, i.e. the images are synchronously recorded.
The stream file parts are saved in the stream folder. To match the constraints set by some file systems, e.g. FAT32, which has a limited file size of 4 GB, the maximum size of a file was set to 2 GB. The stream file parts are stored in a folder tree in the stream folder. The structure of the stream folder and the underlying folders, called *streamfile folders*, is quite similar to the structure of the image folders. The stream folder consists out of up to 1.000 stream file folders named with the convention *szzz* with each $z$ being a single digit. The single character '**s**' is a constant character to show that this folder contains stream file parts. In each stream file folder, the stream file parts are stored. The naming follows the same guidelines as for images, but the file type is *.uuu*. Each folder can contain up to 1.000 stream file parts and in total the system allows up to 1.000.000 stream file parts per project.

### 3.3. StreamReader

Since the structure of the data storage is complex, it is necessary to have suitable interfaces to read the data and access the sensor recordings and images. Therefore a *StreamReader* API was developed, which facilitates the reading of stream files and delivers the information stored in the files, but also provides a simple interface to the images to make the structure transparent again. This API was written in C/C++ for simple integration in common frameworks e.g. OpenCV[7], but was also ported to Python and Matlab for rapid prototyping applications. Another port allows the integration of the StreamReader in ROS.
While reading the stream file, it is possible to set a filter which only returns wanted information while neglecting un-

[7]http://www.opencv.org/

wanted ones. Another possibility is an image-only reading if the interest is focused on visual data, since the image path depends only on the image and camera number.
Besides the API and the datasets, two XML files are needed: one containing all available IDs and messages, their structures and their corresponding IDs and a second one which stores the paths to the dataset and to the first XML file and is used to initialize the StreamReader.

## 4. Data access

The present dataset consists of 7 sequences which have been recorded in different environments, traffic situations and test paths. Two sequences provide several loop closure situations, whereas the others have been recorded on main streets with both low and higher traffic volume. A short description of all 7 sequences is given in Table 3.

Table 3. Recorded sequences

| Name | Length | # Frames | Size | Specifics |
|---|---|---|---|---|
| Seq1 | 0.7km | 4180 | 45 GB | loop closing |
| Seq2 | 2.9km | 9826 | 106 GB | medium traffic |
| Seq3 | 1.4km | 6689 | 72 GB | loop closing |
| Seq4 | 4.6km | 21912 | 236 GB | rush hour |
| Seq5 | ca. 1.4km | 6469 | 70 GB | low traffic |
| Seq6 | ca. 4.3km | 19487 | 210 GB | snow on lens<br>high traffic |
| Seq7 | ca. 9.1km | 39830 | 430 GB | long sequence |

Figure 7 shows example images of the five horizontal cameras (the one pointing upward was disregarded). The images are stored in Bayer format in order to achieve a minimum of redundancy and information loss. Example traffic situations from the dataset are shown in Fig. 8 as panoramic images composed from single images with the Point Grey Ladybug 3 SDK. Figure 6 shows examples of the recordings of the GPS, velocity sensor and IMU (yaw rate) of a similar sequence where the car has been driven three laps around a building.

The APIs for the mentioned programming languages (C/C++, Matlab, Python and as ROS node) and the technical documentation are made available together with the sequences. The documentation contains

- a description of the sequences,
- a detailed sketch of the experimental setup,
- a camera calibration parameter file,
- a detailed description of the stream format,
- a description of the APIs functions and
- a list of IDs and message structures.

All these details are publicly available on our website [8].

[8] http://www.cvl.isy.liu.se/research/datasets/must

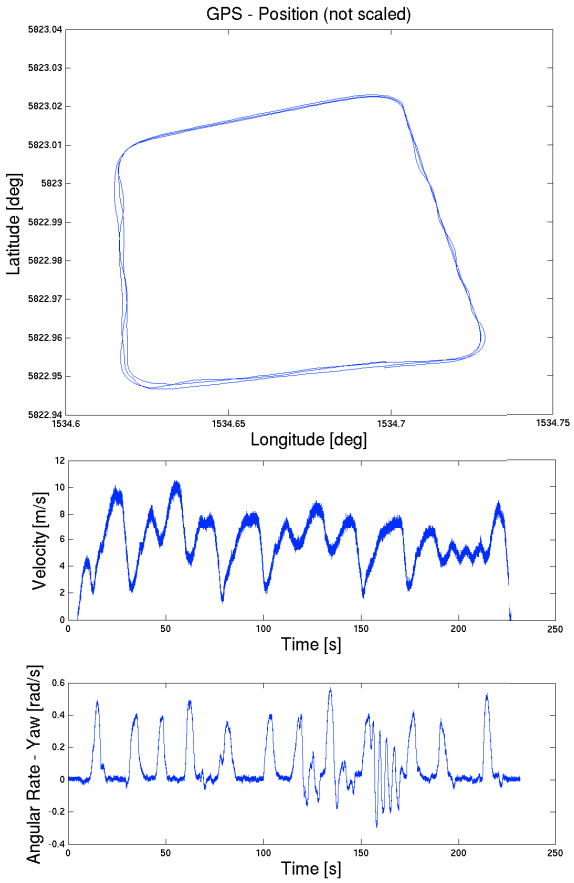


Figure 6. Sensor recordings: GPS recordings (top), Speed data from velocity sensor (middle), Yaw rate from IMU (bottom)

Figure 7. Sample images showing the surrounding of the car

The size of the data recorded so far does not allow us to publish all sequences on a website. However, short sample sequences can be downloaded from the website, and instructions on how to obtain the full dataset are provided there as well.

## 5. Concluding remarks

In this work, we presented two contributions which will help to advance the technology of visual environment sensing for vehicles. The first contribution is the publication

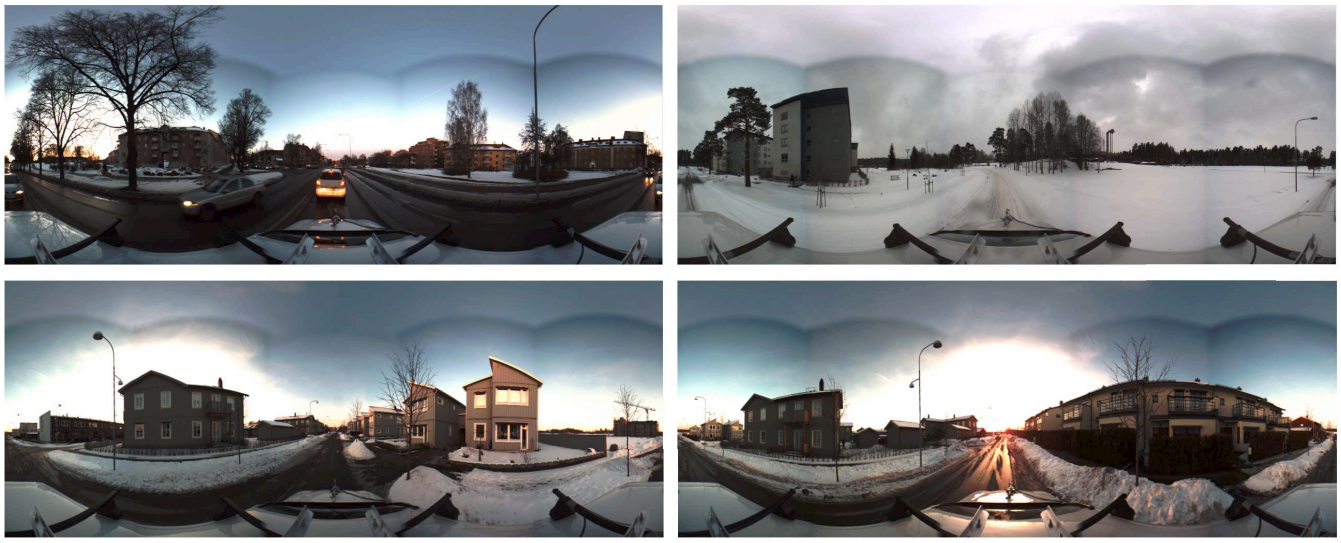

Figure 8. Scenes with different traffic and weather conditions as stitched and clipped panoramic images

of sequences recorded with our experimental car, which allows the community to test and prototype multi-sensor data processing procedures even without having access to an experimental vehicle and the recording hardware. The second contribution is a by-product of our dataset: a versatile multi-purpose dataset API, which allows to record and reproduce asynchronous time stamped multi-sensor data, including multiple cameras.

Benchmark testing of methods that use asynchronous time stamped multi-sensor data can be done in a quasi-standardized way in different research groups. In the long run, this may support the circulation of methods which have been tested under truly realistic situations, and which actually make the leap from an academic development environment into real systems. We hope that other groups will extend the pool of datasets using their vehicles, sensors and cameras. The proposed scheme for organizing the data in general stream files supports the idea that other groups provide similar data in a compatible format.

## Future work

The presented sequences are larger than most existing ones today, but there is a need for even more extensive data and we will record more datasets from diverse real traffic situations in various weather, surrounding and traffic conditions. We also plan, in the future, to increase sensor redundancy in our data for both egomotion and surround sensing, with the aim of allowing a more precise ground truth extraction. Parts of this data, with an emphasis on long sequences, will be made available to the scientific community.

## Acknowledgments

This work has been funded by the Swedish Excellence Center at Linköping - Lund in Information Technology (ELLIIT) and the Linnaeus research environment for Control, Autonomy and Decision-making in Complex Systems (CADICS) at Linköping University.

## References

[1] J.-L. Blanco, F.-A. Moreno, and J. González. A collection of outdoor robotic datasets with centimeter-accuracy ground truth. *Autonomous Robots*, 27(4):327–351, November 2009.

[2] J. Y. Bouguet. Camera calibration toolbox for Matlab, 2008.

[3] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *Computer Vision and Pattern Recognition (CVPR)*, Providence, USA, June 2012.

[4] G. Pandey, J. R. McBride, and R. M. Eustice. Ford campus vision and lidar data set. *International Journal of Robotics Research*, 30(13):1543–1552, November 2011.

[5] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 1, pages I–195. IEEE, 2003.

[6] M. Smith, I. Baldwin, W. Churchill, R. Paul, and P. Newman. The new college vision and laser data set. *The International Journal of Robotics Research*, 28(5):595–599, May 2009.

[7] J.-P. Tardif, Y. Pavlidis, and K. Daniilidis. Monocular visual odometry in urban environments using an omnidirectional camera. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS08)*, 2008.

[8] J. Wiklund, K. Nordberg, and M. Felsberg. Software architecture and middleware for artificial cognitive systems. In *International Conference on Cognitive Systems*, 2010.