ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume IV-1, 2018
ISPRS TC I Mid-term Symposium "Innovative Sensing – From Sensors to Methods and Applications", 10–12 October 2018, Karlsruhe, Germany

DATA PROCESSING AND RECORDING USING A VERSATILE MULTI-SENSOR VEHICLE

Björn Borgmann*, Volker Schatz, Hilke Kieritz, Clemens Scherer-Klöckling, Marcus Hebel, Michael Arens

Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB, Gutleuthausstr. 1, 76275 Ettlingen, Germany
(bjoern.borgmann, volker.schatz, clemens.scherer-kloeckling, marcus.hebel, michael.arens)@iosb.fraunhofer.de

Commission I, WG 6

KEY WORDS: Mobile, Multi-sensor, System, LiDAR, Camera

ABSTRACT:

In this paper we present a versatile multi-sensor vehicle which is used in several research projects. The vehicle is equipped with various sensors in order to cover the needs of different research projects in the area of object detection and tracking, mobile mapping and change detection. We show an example for the capabilities of this vehicle by presenting camera- and LiDAR-based pedestrian detection methods. Besides this specific use case, we provide a more general in-depth description of the vehicle's hard- and software design and its data-processing capabilities. The vehicle can be used as a sensor carrier for mobile mapping, but it also offers hardware and software components to allow for an adaptable onboard processing. This enables the development and testing of methods related to real-time applications or high-level driver assistance functions. The vehicle's hardware and software layout result from several years of experience, and our lessons learned can help other researchers set up their own experimental platform.

1. INTRODUCTION

For many use cases of a mobile sensor system it is preferable to utilize a multitude of different sensor types. This makes it possible to offset the disadvantages of individual sensor types and to combine the specific advantages of the different sensors. For example, a LiDAR sensor is more capable in sensing the exact geometrical structure of the surroundings and it is not dependent on natural light, but it offers lower local data densities than a camera and no color information. Such a mobile multi-sensor system could be a vehicle used for mobile mapping purposes, an autonomous car or a car equipped with high-level driver assistance capabilities.

From a research perspective, it is desirable to have a mobile multi-sensor system which could be used for a wide range of research questions. Therefore, such a vehicle not only has to be equipped with a wide variety of sensors but also needs several other capabilities: For some research questions an efficient system for data recordings is needed. Often it is required that such a system registers the exact recording time and position for each recorded dataset. This is, for example, needed for mapping and change detection related topics. On the other hand, for interactive use cases or live demonstrations onboard processing capabilities are required.

There are several existing vehicle-based multi-sensor systems used by research groups. Typically they are equipped with multiple cameras and LiDAR-sensors. For use cases involving mobile mapping, the cameras are often arranged in a panoramic setup or are part of an integrated panoramic camera head (Anguelov et al., 2010; Paparoditis et al., 2012). Besides mobile mapping purposes, there are systems which are more focused on the development of autonomous driving and driver assistant systems (Geiger et al., 2012). For such use cases the sensor setups often differ to the ones used for mobile mapping. They concentrate more on the recording of the surroundings on street level and the area in front of the vehicle. Typically, additional sensor types like radar are used.

In this paper we present a mobile multi-sensor vehicle which we have designed to fulfill multiple tasks and which is used in different research projects. In contrast to many other mobile mapping systems, our system not only uses cameras and LiDAR sensors, but it also includes a pan-tilt unit which is equipped with multiple sensors. In addition it has the necessary hardware capabilities for an onboard data processing and provides a software environment for such a processing which allows for a standardized and convenient access to the sensors and other hardware components. With regard to the hardware, we based the design of the vehicle on past experiences with the development of an airborne laser scanning system (Schatz, 2008). Currently we use the vehicle for several research topics in the area of object detection and tracking (Becker et al., 2016; Borgmann et al., 2017, 2018; Hammer et al., 2018), mobile mapping and change detection (Gehrung et al., 2018). In the later part of this paper, we present the use case of pedestrian detection, which uses several capabilities of the vehicle.

2. SENSOR SYSTEM

In this section we present our sensor vehicle MODISSA. The vehicle is used for several research projects and serves as a measurement vehicle, experimental vehicle and demonstrator. The main design goal of the vehicle has been the usage in our research environment. Meaning flexibility and the possibility to run experimental software and algorithms are of greater importance than a compact system design. The current sensor setup of the vehicle is determined by our specific needs. Therefore we mostly use cameras and LiDAR sensors while we do not use radar sensors, which are not in our research focus. In the following sections we explain the hardware and software components of MODISSA.

*Corresponding author

2.1 Overview

Since MODISSA serves different purposes and is used for several different research projects, its setup tries to cater for all these use cases. The vehicle is based on a VW transporter which offers enough space to install equipment and to work inside the vehicle with multiple persons. MODISSA and its external components are shown in Figure 1.

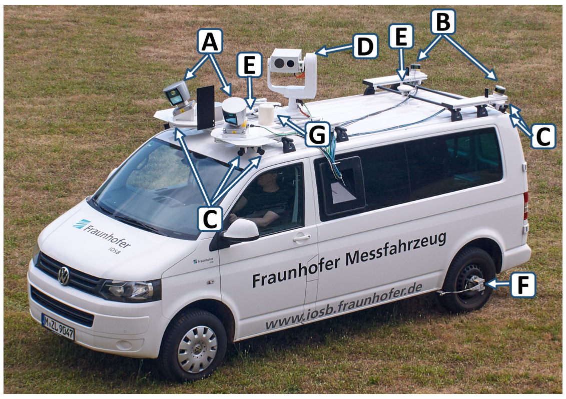

Figure 1. The external components of the sensor system MODISSA
A: 2x Velodyne HDL-64E LiDAR
B: 2x Velodyne VLP-16 LiDAR
C: Panoramic camera setup 8x Baumer VLG-20C.I. 2 cameras per corner
D: Pan-tilt unit with Jenoptik IR-TCM 640 thermal infrared camera, JAI CM 200-MCL gray scale camera and Jenoptik DLEM 20 laser rangefinder
E and F: Applanix POS LV V5 520 inertial navigation system with GNSS antennas (E), distance measuring indicator (DMI) (F), inertial measurement unit (IMU) and position computer (both not visible)
G: External WiFi antenna

The vehicle is in total equipped with 4 LiDAR sensors. Two high resolution Velodyne HDL-64E are mounted at the front and two lower resolution Velodyne VLP-16 at the back. Both kinds of LiDAR sensors use a rotating head and therefore have a horizontal field of view of $360°$. The vertical field of view of the HDL-64E is from $2°$ to $-24.33°$, which is covered by 64 scan lines. The VLP-16 have a field of view from $15°$ to $-15°$ covered by 16 scan lines. The HDL-64E at the front are currently tilted by $25°$ towards the diagonal away from the vehicle to prevent measurements of the vehicle’s roof. This setup also allows the LiDAR sensors to scan most of the outside walls and partly the roofs of buildings, which is useful for mobile mapping purposes. The tilt angle of the back VLP-16 LiDAR sensors is $15°$. Considering the vertical field of view of the sensors, this setup basically changes their field of view to $0°$ to $-30°$. So these sensors mostly cover the ground level and increase the ability to perceive the complete surroundings of the vehicle. For example, the LiDAR sensors are used to detect pedestrians in the surroundings of the vehicle, which is explained in Section 3.1.

On each corner of the vehicle two cameras are mounted. All eight of these cameras form a panoramic camera setup which covers the complete surroundings of the vehicle. In the past we have experimented with using a single integrated panoramic camera mounted on the top of the vehicle to achieve the same result. But that setup was problematic since the camera either obstructed the field of view of other sensors or was itself obstructed by them. Mounting the cameras at the corners has solved these problems and gives each camera an unobstructed field of view. The disadvantage of this setup is that the cameras do not share a projection center, which makes it more difficult to stitch the images together to form a $360°$ panorama. In addition to the LiDAR sensors, these cameras are used for a pedestrian detection in the vehicle surroundings, which is explained in Section 3.2.

On the center position of the vehicle’s roof a pan-tilt unit is mounted. This unit is gyro stabilized and can be aimed at a certain geo coordinate while the vehicle is moving. It is equipped with a thermal infrared camera, a gray scale camera for visible light and a laser rangefinder. The laser rangefinder allows us to determine the distance to a point roughly on the optical axis of the cameras on the pan-tilt unit. The infrared camera has a wide field of view of $65.2°$ x $51.3°$ and it operates in a spectral range from $7.5\,\mu m$ to $14\,\mu m$. Since it is able to perform thermal imaging, it could also work as a night vision device. In comparison to the other cameras, the camera for visible light has a smaller field of view of $25.2°$ x $19.3°$ and is mainly used for optical sensing in greater distances to supplement the panoramic cameras and the infrared camera.

The vehicle is equipped with an inertial navigation system (INS), which is used to determine the position of the vehicle. In comparison to a simple GNSS system, an INS allows us to constantly determine the vehicle’s position in a high temporal and geometrical precision. This makes it possible to perform a direct georegistration of the recorded data, which is used to create consistent point clouds from LiDAR measurements even if the vehicle is moving. This process is explained in Section 2.6. The INS is also used to provide a central common clock for the complete sensor system and to synchronize the sensors as well as other components (cf. Section 2.2). The INS is a Applanix POS LV V5 520 which consists of several separate components: An inertial measurement unit (IMU), two GNSS antennas, a distance measuring indicator (DMI) which is mounted on a wheel of the vehicle, and a processing unit. The second GNSS antenna helps to determine the heading of the vehicle, even if it is not moving. According to the manufacturer, in ideal circumstances the INS has an accuracy of up to $3.5\,\mathrm{cm}$ for the X,Y position, $5\,\mathrm{cm}$ for Z position (height axis) and a rotational accuracy of $0.008°$ for roll and pitch and $0.02°$ for the heading (RTK). Unfortunately in real use cases the actual position accuracy is often far below these values.

A rack which is equipped with several PCs and further electronic components is mounted inside. Three of the PCs are mainly used for the interaction with the sensors, data recording and tasks in the area of sensor synchronization and controlling of the pan-tilt unit. Two additional PCs are mainly used for the onboard data processing, each equipped with two powerful multi-core Intel Xeon CPUs and additionally GPUs. The onboard processing is further explained in Section 2.5. The rack is mounted in the passenger area of the vehicle and replaces the middle row of seats. In the trunk of the vehicle, components are mounted to provide electrical power for the other components. To achieve this, large lithium-ion batteries and a power inverter are used. Therefore the electrical power is provided completely independently of the vehicle’s engine. Up to 2000 W at 230 V AC can be provided continuously for more than five hours before recharging.

Various components of the vehicle are connected with each other over several ethernet networks. There is a general network in which all of the PCs and several other components participate.

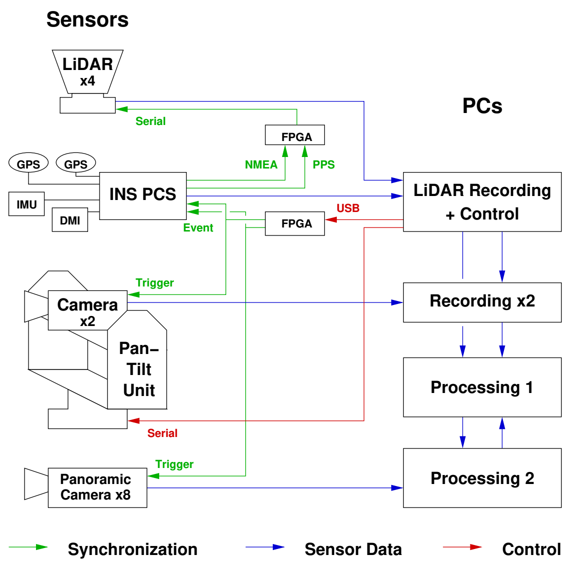

Figure 2. Simplified schematic of data paths for synchronization, data acquisition and control. Sensors operated the same way are only represented once.

This network also includes a WiFi connection, which, for example, is used for demonstration purposes to enable visualizations on external presentation displays. An additional network is used to transfer the data from the LiDAR sensors to a recording and a processing PC. A third network directly connects the two processing PCs with a 10 GBit high-bandwidth connection. This connection was added to deal with the high data transfer requirements of some onboard processing use cases.

2.2 Sensor synchronization

One field of applications for which the sensor system is used can benefit from data fusion techniques. That requires a matching of data from different sensors recorded at the same time. To that end, accurate timestamping of all imaging data was designed into the system. Figure 2 shows an overview of synchronization signals as well as sensor data transmission and control.

Precise hardware synchronization requires a central common clock. In our system, this clock is provided by the INS. Its position computer (PCS) provides four event inputs for timestamping signal pulses and four serial interfaces that can transmit National Marine Electronics Association (NMEA) messages and associated pulses. Both were used for different kinds of sensors.

The LiDAR sensors are designed to synchronize their internal clock by receiving an NMEA message and a pulse-per-second (PPS) signal. The latter indicates the start of the second of the following NMEA message. Unfortunately the LiDAR sensors demand a type of NMEA message that the INS does not support and specific relative timing of the PPS pulse and the message, so they cannot be connected directly to the INS. Interface logic that translates the message and ensures the appropriate timing was developed at our institute and implemented in a small Field-Programmable Gate Array (FPGA). It is present in the system on a GODIL board from OHO Elektronik mounted in a case and powered via a USB cable but otherwise operating autonomously. The board carries a Xilinx Spartan 3E FPGA, flash RAM for storing its configuration and a microcontroller translating from USB to a serial port connected to the FPGA.

All cameras are operated in external frame trigger mode, i. e., every single frame is triggered by a signal external to the cameras. Trigger signals are generated by another custom logic design implemented on an FPGA. The FPGA board used is the same as the one for interfacing between the INS and the LiDAR sensors. The trigger generator design supports separate trigger channels for the video, microbolometer and panoramic cameras. The trigger outputs are duplicated, with one output leading to the cameras and the other to one of the event inputs of the INS, as shown in Figure 2. That causes every triggered frame to be timestamped. The timestamps from the INS data are embedded into the camera frame data streams after data acquisition as described in the following section.

The USB interface of the trigger generator FPGA board is attached to one of the PCs. It provides a simple textual register interface that allows for changing frame intervals and starting and stopping trigger operation. Commands are transmitted to the FPGA via the microcontroller on the FPGA board.

2.3 Data acquisition

Data from the various sensors are acquired by the different PCs mounted in the vehicle interior. The data acquisition software was developed at our institute and allows us to optionally record the data to disk while also making it available for online processing. The details of how this is achieved differ between sensors.

The LiDAR sensors broadcast their data over their network connection via the User Datagram Protocol (UDP). A network switch connects them to the PC recording LiDAR data and one of the processing PCs. The data are thereby available for recording and online processing simultaneously without any special effort. In order to limit network traffic and ensure uninterrupted data transmission, it was necessary to configure the switch so that packets from the LiDAR sensors are only forwarded to the PCs, not to each other. That was achieved by creating a separate Virtual Private Network (VPN) for each LiDAR and PC and selectively enabling forwarding between them. The PCs receive the data via listening sockets bound to the ports to which the LiDAR sensors transmit. Position and time data that some of the LiDAR retransmit from the synchronization with the INS are received on a different port and are also recorded.

Due to the relatively small data rate of the LiDAR sensors, the PC recording LiDAR data is also used for other tasks, as can be seen in Figure 2. It runs a server program controlling the pan-tilt unit that implements automated tracking and receives commands from other components of the system. The same PC also receives two navigation data streams from the INS. One is a buffered stream that is recorded to disk. The other is a low-latency real-time stream that is made available to other system components by a custom navigation data server. The navigation data server is used by the pan-tilt unit control server for tracking a fixed location. It also provides navigation data and pan-tilt angles for embedding into the camera data streams (see below).

Data from the cameras are acquired by suitable frame grabbers. Each camera type is attached to a different PC. The video and infrared cameras on the pan-tilt unit have small dedicated recording PCs, while the panoramic cameras are attached to one of the processing PCs. The data are either written to a file on disk or to a ring buffer in a shared memory region as shown in Figure 3. Both can be accessed via memory-mapping by programs processing them further. On the recording PCs for the pan-tilt

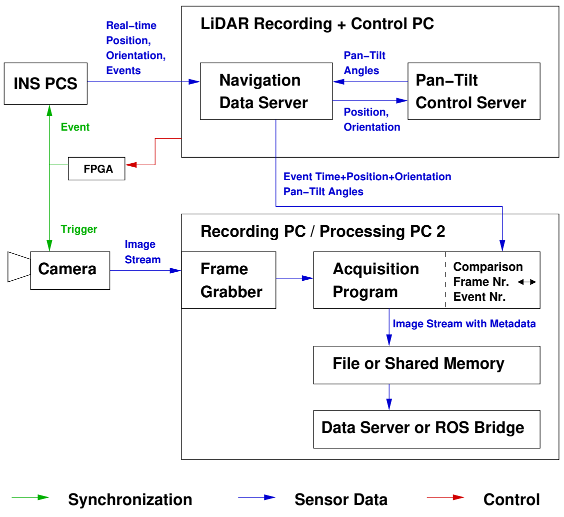

Figure 3. Schematic of data acquisition from cameras and embedding of metadata into image streams. The navigation data server interpolates the position and orientation to the event (trigger) times. See text for explanation.

cameras, two server programs make the acquired data available for online processing and visualization, respectively. The former provides the uncompressed data to other components of the system via a Transmission Control Protocol (TCP) network socket. The visualization server compresses the image data to a low-bandwidth OGG/Theora stream and also transmits it via TCP. Due to their high total data rate of approximately 150 MBytes/s, the panoramic camera data cannot be easily transmitted over the network. Access to their data is provided on the processing PC to which they are attached, as described in Section 2.5.

Data from the cameras are recorded in an uncompressed format developed at our institute, the Streaming Image Sequence (SIS [1]) format. Its design requirements were losslessness, permitting multiple channels and various numerical pixel data types, allowing for optional metadata blocks in the stream and frame headers, and supporting automatic endian conversion. No pre-existing format fulfilled all those requirements. The SIS format does not support compression, but the whole stream could be compressed with a standard method if and when that is considered advantageous.

The camera data acquisition programs embed metadata transmitted by the navigation data server into optional blocks in each frame header of their image data stream. As a result, both online and offline processing tasks can work on frames with associated metadata. The metadata comprise the frame timestamp, the position and orientation at the frame time and the current angles of the pan-tilt unit. Figure 3 shows how the embedding works. The position and orientation are interpolated by the navigation data server to the frame times from regular navigation solution packets received from the INS. The data acquisition programs synchronize the metadata and frame data streams by detecting gaps in their sequential indices and skipping the respective other stream. When starting data acquisition, a small delay is introduced before starting camera triggering, to make sure the first triggered frame is recorded and therefore matches the first trigger timestamp.

2.4 Pan-tilt unit control and tracking

As was mentioned in the previous section, the pan-tilt unit is controlled by a server program running on the PC that also records LiDAR and navigation data. This server accepts commands to aim the cameras on the pan-tilt unit at a direction relative to the vehicle, at a compass direction or at a geographical location and elevation. For the latter two commands, the pan-tilt server uses real-time navigation and orientation data from the INS retransmitted by the navigation data server (see Figure 3) to track the direction or location continuously.

The pan-tilt unit, a General Dynamics Vector 50, is controlled in such a way that movement of the vehicle is smoothly compensated, even when driving on unpaved roads or on the lawn at our institute. It supports internal gyro stabilization, which is advantageous due to its shorter control loop. However, when the gyro stabilization is enabled, its maximum rotational speed is reduced by a factor of four. In order to avoid this tradeoff, our control server adaptively switches between these modes depending on the situation. When fast turning is required, such as when driving past close to the target location or when changing the target location, the stabilization is disabled, otherwise it is enabled. In those situations, camera frames tend to be degraded by motion blur, so the less accurate stabilization performed by the control server based on INS data is acceptable.

2.5 Onboard processing

To cover certain research questions in which a direct interaction with the vehicle is required and for the performing of live demonstrations, the vehicle has some onboard processing capabilities. As mentioned earlier, for this purpose mainly two PCs are used. In regard to the software environment, we had several requirements to allow for an efficient development, evaluation and demonstration of methods and processing capabilities. We needed the ability to exchange data between several software processes which potentially run on different PCs. Therefore this data exchange has to utilize network connections which should be transparent for the actual data processing. In addition, the data exchange should be able to deal with multiple data receivers. We also needed a way to deal with several coordinate frames which exist in context of the data processing. Each sensor has its own coordinate frame, the vehicle itself has a reference frame, and a global coordinate frame is needed to consider the vehicle movement in the world. Many of the transformations between these coordinate frames are fixed and defined by the construction of the sensor system, but due to the movement of the vehicle and the pan-tilt unit, several transformations are constantly changing. Therefore, a way is required to provide at least these constantly changing transformations to the data processing. It is also desirable to emulate the onboard environment to support the development of software that is intended to run on the vehicle.

To fulfill our requirements, we decided to use ROS (Robot Operating System[2]). Although ROS has originally been designed for the usage with robots, it covers our requirements. In the context of ROS a program is called *node* and all running nodes in a ROS system form a network which is called the ROS computation graph. The nodes in such a graph can be distributed among an arbitrary amount of PCs, which is transparent for the user as long as they are able to communicate with each other over a network connection. Nodes can send and receive ROS messages which are published on so-called *topics*. This allows for a communication between nodes in a sender-receiver format and is used for continuous data transfers like the transfer of currently recorded sensor data. Such messages usually have some header information like

[1]http://s.fhg.de/sis

[2]http://www.ros.org

a timestamp or coordinate frame. Every node can subscribe to a topic to receive the messages which are sent on it and each topic can be subscribed by an arbitrary amount of nodes. ROS services are a second way of communication between nodes, which is used in a request-response format to request a certain function or data from another node. For both ways of communication, standard data types are provided which already cover many use cases. Additional data types can be created if needed. The complete communication via messages or a subset of this communication can be recorded in so-called *bag* files which later can be played back. These bag files fulfill our requirement to be able to emulate the onboard environment.

ROS makes it possible to create, fill and access a transformation tree which provides necessary transformations between various coordinate frames. This tree can contain fixed transformations as well as dynamic ones. The dynamic transformations usually have a timestamp and it is possible to retrieve "historic" transformations. These capabilities are used to determine the exact state which the tree had at the time of acquisition of a certain dataset. ROS also provides many libraries and tools for data processing, visualization and debugging. Especially the visualization components are helpful for the fast prototyping of a demonstrator or the debugging of programs, since it is possible to visualize many standard data types which are transmitted as ROS messages.

In our ROS system we provide an abstraction layer for the receiving of sensor data and access to other hardware components of the vehicle. This abstraction layer is comprised of at least one ROS node for each sensor. We call these nodes *bridge nodes* and their task is to receive data from the sensors and provide them to other nodes. For the cameras which are mounted on the pan-tilt unit, nodes receive their data via the network from their respective data acquisition PCs in form of uncompressed data streams in SIS format. The data from the panoramic cameras are also received in the SIS format, but since these cameras are already connected to one of the processing PCs and to reduce network traffic, we read from the recorded file or shared memory directly without a network transmission.

The SIS format already contains the exact timestamp in the frame header, as described in Section 2.3. This timestamp is used to fill the according field in the ROS message headers. For the panoramic cameras, the abstraction layer contains an additional node which receives the raw data from the bridge node and converts it from its original Bayer pattern format into an RGB format. For the infrared camera, a similar node performs a conversion from 16-bit color depth, of which most of the times only an interval of about 12 bits is actually used, into 8-bit color depth. This conversion is done by determining the actually used data interval and mapping it linearly onto the full 8-bit range. Such a data format is better suited for visualization and often helpful for further processing. Since each ROS topic can be received by multiple nodes, a "data consumer" can still use the raw data from the bridge nodes if desired.

For the Applanix INS system, the abstraction layer also consists of two nodes. One receives the data from the Applanix system and sends it into the ROS environment (unchanged, in terms of the content). The other uses these data to add a transformation to the transformation tree, which converts between the vehicle reference coordinate frame and an ECEF (Earth Centered Earth Fixed) global coordinate frame. This transformation is based on the current vehicle position provided by the INS. An additional constant transformation between the ECEF coordinate frame and a local ENU (East North Up) coordinate frame is provided by this node. The origin of that coordinate frame is determined once at system start either based on a configuration parameter or the current vehicle position. We often use such an ENU coordinate frame with a local origin instead of ECEF as a Cartesian world coordinate frame. It has the advantage that the coordinate values are lower, allowing the usage of single precision data types. Our point cloud processing and ROS as well normally use point clouds with single precision coordinates. For point clouds in a world coordinate frame, we use this ENU coordinate frame. It also offers a local vertical axis which is helpful for certain processing algorithms.

The LiDAR sensors provide their data via the network and already embed an exact acquisition timestamp into the data, which is used for the ROS message header. This timestamp is based on the data provided to them by the INS (cf. Section 2.2). The bridge nodes for the LiDAR sensors receive their raw data via the network and transfer them into the ROS environment. A second group of nodes uses these data and the INS data to generate point clouds. These clouds are generated for each sensor in two variants: One refers to the coordinate frame of the sensor and a second one refers to the mentioned local ENU coordinate frame. The generation of point clouds is further explained in Section 2.6.

Additional ROS nodes are provided for controlling the pan-tilt unit from the ROS environment and for triggering a measurement by the laser rangefinder and receiving its result. When needed, a measurement of this sensor can be triggered via a ROS service call. The pan-tilt unit is also controlled by ROS services. The node which provides these services itself uses the server program described in Section 2.4 and translates its functionality into ROS services. The node also receives the current orientation of the pan-tilt unit from this program and publishes it into the ROS transformation tree to allow transformations between coordinate frames of sensors mounted on the pan-tilt unit and other coordinate frames.

After the described hardware abstraction layer, further ROS nodes perform the processing that is specific for the current use case. This could be single nodes or a group of nodes which interact with each other. Since a node is a normal program which uses some ROS specific libraries for the interaction with the ROS environment, it is easy to convert arbitrary programs into ROS nodes. Such a program can use all the capabilities provided by our hardware abstraction layer and is able to access all sensors and further hardware components of our vehicle without the need to implement such a hardware interaction itself.

2.6 Processing of LiDAR data

We use scanning LiDAR sensors with a rotating head, which we operate at a rotation rate of 10 revolutions per second. For further processing of the data acquired by these sensors, we normally prefer to use point clouds containing one rotation of the scan head instead of raw distance measurements. As the reference frame for these point clouds we use an ENU coordinate frame with a local origin. When the vehicle is moving, we have to consider this movement as part of the point cloud generation. The process which we use for the point cloud generation is performed for each LiDAR sensor separately and is identical for the offline processing of recorded data and for the onboard processing of live data. The only difference is the source of the data to be processed. For the offline processing, we use files with the recorded raw LiDAR and the INS data. These files can be synchronized based on timestamps which are part of both files. For the onboard processing, we use the data streams provided by our bridge nodes for the INS and the LiDAR sensors. In case of the offline processing, we normally perform a postprocessing of the INS data using data from public GNSS reference stations (SAPOS). This increases the precision of the INS data. In the live processing such reference data are not used.

Based on the timestamps embedded in the LiDAR data and the specifications of the LiDAR sensors itself it is possible to determine the point in time for each distance measurement. This information is used to determine the exact position and orientation of the vehicle at the time of each measurement based on the data from the INS. Our INS delivers position and orientation at a rate of 200 Hz. Since the LiDAR performs measurements at a higher rate, we use a linear interpolation to cover the gaps between two INS measurements. Although such a simple linear interpolation has its limits, this approximation is sufficient since we only interpolate over the short time periods between two INS measurements. Besides the INS information, we use intrinsic calibration parameters for each LiDAR and extrinsic calibrations between the LiDAR and INS reference frame for the generation of point clouds.

3. USE CASE: PEDESTRIAN DETECTION

In this section, we present the detection of pedestrians in the surroundings of the vehicle as a use case for our multi-sensor vehicle. In the area of autonomous driving and driver assistance systems, such a functionality is of great interest. But it could also be part of a mobile mapping system to remove pedestrians as unwanted noise from the recorded data or to anonymize them for privacy reasons.

We perform a pedestrian detection with different sensor types. This allows us to profit from the individual advantages of the different sensors and to compensate their disadvantages. For example, a LiDAR sensor is able to directly and very accurately determine the geometric structure of the scene and the 3D position of the detected pedestrians. On the other hand, a camera operates with a higher data density which increases the detection performance. It also makes it possible to detect additional features such as the faces of persons, which can be used to estimate their viewing direction. In case of low-light situations, we plan to use the thermal infrared camera mounted on the pan-tilt unit of the vehicle to gather such features instead of the cameras for visible light. In such situations the LiDAR sensors are to be used to point the pan-tilt unit in the direction of pedestrians.

3.1 LiDAR-based pedestrian detection

There are several approaches for the detection of pedestrians or, more generally, objects in 3D LiDAR data. Often this task is divided into a segmentation step and a classification of the segments. Such a segmentation can be achieved by region growing. Different kinds of classifiers are typically used for the classification. One group of such classifiers are support vector machines (SVM) which fit a hyperplane in feature space to differentiate between different classes. They are used by Navarro-Serment et al. (2010) for the detection and tracking of persons in LiDAR point clouds. Another group of classifiers are bag-of-words approaches. They use words described by feature descriptors. These words fill a dictionary which is the result of a training process. Feature descriptors for the processed data are generated while classifying data. These descriptors are matched to words in the dictionary which then vote for a classification of the data. Behley et al. (2013) combine several bag-of-words classifiers for the classification of point cloud segments. These classifiers use differently parameterized features which allows them to deal better with the characteristics, such as the data density of each processed segment. Recently, deep convolutional neural networks have been used for object recognition tasks in 3D data. To achieve this, the data are either converted into depth images (Socher et al., 2012) or into a volumetric representation (Maturana and Scherer, 2015; Garcia-Garcia et al., 2016).

Our own approach is based on 3D Implicit Shape Models (ISM) and we presented it in greater detail in Borgmann et al. (2017). ISM are an extension of the bag-of-words approach in which words not only vote for a class but also for the position of that class. Therefore they not only consider the existence of certain features but also their geometrical distribution. They have been used for several object recognition tasks in 3D data (Velizhev et al., 2012; Knopp et al., 2010). Our own ISM approach is modified in a way to better deal with and profit from the usage of multiple LiDAR sensors in parallel (Borgmann et al., 2018). On our vehicle MODISSA the multiple LiDAR sensors have some overlaps in their coverage. Since the sensors are rotating scanners, it is nearly impossible to synchronize them in a way that they scan same areas at the exact same time. In case of moving objects there will be some movement induced distortion effects if the point clouds of multiple sensors are directly merged. To be able to profit from the higher data density and better coverage in such areas, we merge the data in the voting space of the ISM method. In this space the method already deals with uncertainties and can accept additional uncertainties caused by the distortion effects without reducing the overall performance of the method. Hence, at first we perform a feature extraction. This is followed by a search for matching words and the cast of votes for each sensor. Then we perform the final ISM steps for all sensors together, the search for maxima in the voting space. In addition, our method does not rely on a data segmentation which prevents problems caused by segmentation errors.

3.2 Image-based pedestrian detection

Pedestrian detection in images is a key component in many computer vision applications, e.g. human-computer interaction, advanced driver-assistance systems, and surveillance. One of the popular early approaches is Histograms of Oriented Gradients (HOG) (Dalal and Triggs, 2005). The approach calculates handcrafted features over the whole image and then applies a trained classifier in a sliding window manner. A non-maximum suppression is then used to group positive classifications to the final detections. A similar pipeline is applied in later pedestrian detection methods based on integral channel features (Dollár et al., 2009, 2014; Benenson et al., 2012), which have been shown to work also in infrared images (Kieritz et al., 2013).

In recent years, detectors based on deep convolutional neural networks have gained popularity (R-CNN (Ren et al., 2015), YOLO (Redmon et al., 2016), SSD (Liu et al., 2016)). While R-CNN uses a region proposal network followed by a classifier neural network, YOLO and SSD adopt an object classifier and a bounding box regression in the last convolution layers of a single object detection network.

For our sensor vehicle MODISSA, we adapt the SSD method because of its low runtime (see Figure 4 for detection examples on the panoramic cameras). We trained the network for a 480x270

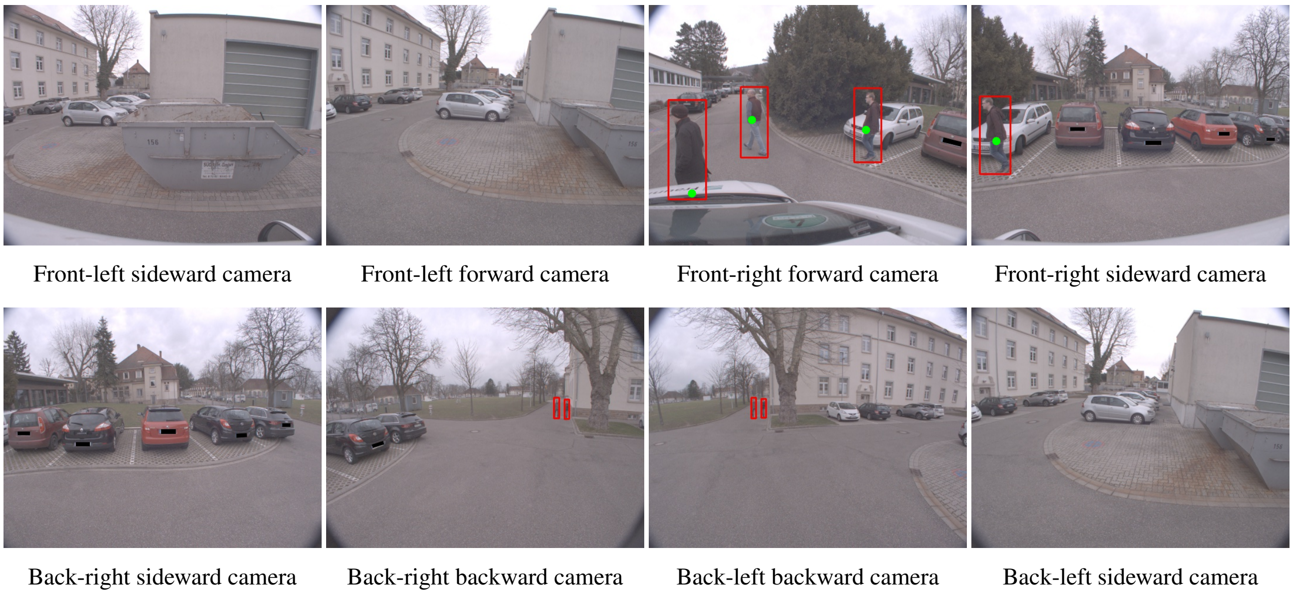
Front-left sideward camera
Front-left forward camera
Front-right forward camera
Front-right sideward camera
Back-right sideward camera
Back-right backward camera
Back-left backward camera
Back-left sideward camera

Figure 4. Pedestrian detection of five people using the panoramic cameras (red bounding boxes) and LiDAR-sensors (green dots) of MODISSA.

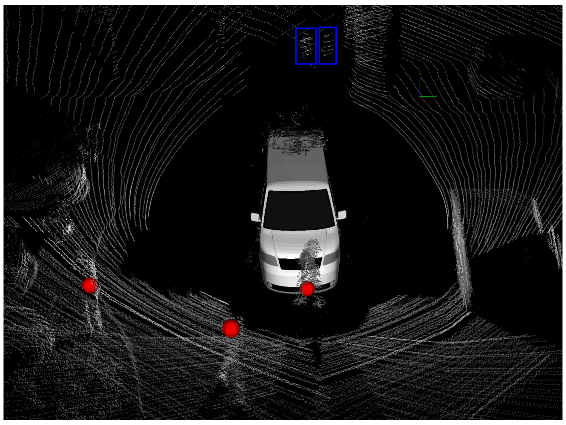

Figure 5. Pedestrian detection using the front LiDAR sensors of MODISSA. Red dots are correct detections; blue rectangles are missing detections. The scene corresponds to Figure 4 and shows the same moment of the same measurement run.

input image size to detect pedestrians. It was trained for 300.000 iterations using the following pedestrian detection datasets: Caltech pedestrian detection benchmark (Dollár et al., 2012), INRIA test set (Dalal and Triggs, 2005), and CityPersons training set (Zhang et al., 2017).

We tested our method on recorded data using a NVIDIA TITAN X graphic card. Each image is scaled to the network input size. When detecting pedestrians in all eight camera images in parallel, we achieve an average processing speed of 5 frame sets per seconds (197 ms). When processing one single camera, we achieve an average runtime of 28.5 frames per second (35 ms).

3.3 Preliminary results

We are currently still in the process of evaluating both our pedestrian detection approaches, LiDAR-based and camera-based. In this paper we present some preliminary results. For the evaluation of our methods we use different scenarios with various levels of difficulty. The easiest ones have been recorded with a stationary vehicle on an empty field while the more difficult ones use a moving vehicle in a street environment with several parked cars, vegetation and other obstacles. Figure 4 and Figure 5 exemplarily show detection results for the same point in time, achieved with the panoramic cameras and the front LiDAR sensors of our vehicle.

The results show that the panoramic camera setup works well for the detection of pedestrians in the complete surroundings of the vehicle. It also shows that there are several overlapping areas in the fields of view of the cameras which we have to consider in a later data fusion method to determine the amount of unique persons in the surroundings. The LiDAR sensors provide quite detailed position information for the detected persons. This is shown in Figure 5 for the detected persons in front of the vehicle. But we have some problems with detecting them in greater distances, especially if they are only partly visible. This is shown by the two persons far behind of the vehicle. Our current LiDAR setup is not optimally suited for this use case. The setup is optimized for mobile mapping purposes and limits the ability of the LiDAR sensors to detect pedestrians in greater distance in front of the vehicle and directly behind it. This can be improved in the future by using a different tilt angle for the front LiDAR sensors.

4. CONCLUSION

We have presented a multi-purpose sensor vehicle which can be used for data recording as well as real-time applications. The vehicle’s hardware and software setup result from several years of experience, and our lessons learned can help other researchers set up their own experimental platform. The vehicle is equipped with several cameras for the visible and the infrared spectrum and multiple LiDAR sensors. It also has additional components like an inertial navigation system and a pan-tilt unit. The vehicle has two general operation modes: One provides a data recording which records useful metadata besides the actual data to improve the offline usage of the recorded data. The other mode is onboard processing of the data being acquired. This mode is based on the ROS middleware and allows us to develop, test and demonstrate a multitude of methods directly on the vehicle. Therefore the vehicle is able to cover various research questions. We have demonstrated these capabilities by presenting a use case in which we utilize the vehicle in a current research topic, the detection of pedestrians.

In future work, some modifications of the orientation of the LiDAR sensors are planned to achieve a setup which is better suited for pedestrian detection. We also plan to integrate the infrared camera in this use case and to evaluate the performance in low-light scenarios in which the LiDAR sensors could be used to point the thermal infrared camera in the direction of detected persons. In addition we work on several other use cases, such as mobile mapping and change detection.

References

Anguelov, D., Dulong, C., Filip, D., Frueh, C., Lafon, S., Lyon, R., S. Ogale, A., Vincent, L. and Weaver, J., 2010. Google street view: Capturing the world at street level. 43, pp. 32–38.

Becker, S., Scherer-Negenborn, N., Thakkar, P., Hübner, W. and Arens, M., 2016. The effects of camera jitter for background subtraction algorithms on fused infrared-visible video streams.

Behley, J., Steinhage, V. and Cremers, A. B., 2013. Laser-based segment classification using a mixture of bag-of-words. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 4195–4200.

Benenson, R., Mathias, M., Timofte, R. and Van Gool, L., 2012. Pedestrian detection at 100 frames per second. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2903–2910.

Borgmann, B., Hebel, M., Arens, M. and Stilla, U., 2017. Detection of persons in MLS point clouds. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2/W7, pp. 203–210.

Borgmann, B., Hebel, M., Arens, M. and Stilla, U., 2018. Usage of multiple LiDAR sensors on a mobile system for the detection of persons with Implicit Shape Models. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLII-2, pp. 125–131.

Dalal, N. and Triggs, B., 2005. Histograms of oriented gradients for human detection. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Vol. 1, pp. 886–893.

Dollár, P., Appel, R., Belongie, S. and Perona, P., 2014. Fast feature pyramids for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36(8), pp. 1532–1545.

Dollár, P., Tu, Z., Perona, P. and Belongie, S., 2009. Integral Channel Features. In: *Procedings of the British Machine Vision Conference*.

Dollár, P., Wojek, C., Schiele, B. and Perona, P., 2012. Pedestrian detection: An evaluation of the state of the art. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(4), pp. 743–761.

Garcia-Garcia, A., Gomez-Donoso, F., Garcia-Rodriguez, J., Orts-Escolano, S., Cazorla, M. and Azorin-Lopez, J., 2016. Pointnet: A 3D convolutional neural network for real-time object class recognition. In: *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 1578–1584.

Gehrung, J., Hebel, M., Arens, M. and Stilla, U., 2018. A voxel-based metadata structure for change detection in point clouds of large-scale urban areas. In: *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. IV-2, pp. 97–104.

Geiger, A., Lenz, P. and Urtasun, R., 2012. Are we ready for autonomous driving? The KITTI vision benchmark suite. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hammer, M., Hebel, M., Borgmann, B., Laurenzis, M. and Arens, M., 2018. Potential of LiDAR sensors for the detection of UAVs.

Kieritz, H., Hübner, W. and Arens, M., 2013. Learning transmodal person detectors from single spectral training sets. In: *Optics and Photonics for Counterterrorism, Crime Fighting and Defence IX; and Optical Materials and Biomaterials in Security and Defence Systems Technology X*, Vol. 8901, International Society for Optics and Photonics.

Knopp, J., Prasad, M., Willems, G., Timofte, R. and Van Gool, L., 2010. Hough transform and 3D SURF for robust three dimensional classification. In: *Proceedings of the 11th European Conference on Computer Vision: Part VI*, ECCV'10, Springer-Verlag, Berlin, Heidelberg, pp. 589–602.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y. and Berg, A. C., 2016. SSD: Single shot multibox detector. In: *European conference on computer vision*, Springer, pp. 21–37.

Maturana, D. and Scherer, S., 2015. Voxnet: A 3D convolutional neural network for real-time object recognition. In: *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 922–928.

Navarro-Serment, L. E., Mertz, C. and Hebert, M., 2010. Pedestrian detection and tracking using three-dimensional LADAR data. *The International Journal of Robotics Research* 29(12), pp. 1516–1528.

Paparoditis, N., Papelard, J.-P., Cannelle, B., Devaux, A., Soheilian, B., David, N. and Houzay, E., 2012. Stereopolis II: A multi-purpose and multi-sensor 3D mobile mapping system for street visualisation and 3D metrology. 200, pp. 69–79.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You only look once: Unified, real-time object detection. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779–788.

Ren, S., He, K., Girshick, R. and Sun, J., 2015. Faster R-CNN: Towards real-time object detection with region proposal networks. In: *Advances in neural information processing systems*, pp. 91–99.

Schatz, V., 2008. Synchronised data acquisition for sensor data fusion in airborne surveying. In: *2008 11th International Conference on Information Fusion*, pp. 1–6.

Socher, R., Huval, B., Bath, B., Manning, C. D. and Ng, A. Y., 2012. Convolutional-recursive deep learning for 3d object classification. In: *Advances in Neural Information Processing Systems*, pp. 656–664.

Velizhev, A., Shapovalov, R. and Schindler, K., 2012. Implicit shape models for object detection in 3D point clouds. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* I-3, pp. 179–184.

Zhang, S., Benenson, R. and Schiele, B., 2017. Citypersons: A diverse dataset for pedestrian detection. In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3213–3221.