Design and Implementation of a LIDAR Based Range Sensor System

Stefan Hensel(1), Marin B. Marinov(2), Markus Obert(1) and Dimitre Trendafilov(2)
(1)University of Applied Sciences Offenburg, Department for Electrical Engineering, Offenburg, Germany
(2)Technical University of Sofia, Department of Electronics, Sofia, Bulgaria

***Abstract*— Positioning mobile systems with high accuracy is a prerequisite for intelligent autonomous behavior, both in industrial environments and in field robotics. This paper describes the setup of a robotic platform and its use for the evaluation of simultaneous localization and mapping (SLAM) algorithms. A configuration using a mobile robot Husky A200, and a LiDAR (light detection and ranging) sensor was used to implement the set-up. For verification of the proposed setup, different scan matching methods for odometry determination in indoor and outdoor environments are tested.**

***Keywords*— LiDAR, LOAM (Lidar Odometry and Mapping), mapping, machine learning, mobile robotics, navigation, ROS, SLAM.**

I. Introduction

The range of areas in which autonomous mobile systems are used is growing continuously. The use of autonomous transport vehicles in industry or robots in private households has now become standard. The development of autonomous motor vehicles is also progressing steadily. Machine learning offers the field of robotics a set of tools for designing difficult and complex behaviors. On the other hand, the challenges of robotics-related problems also provide a positive impact on developments in robot learning.

In all use cases, the basic challenge is more or less the same. Autonomous mobile systems must simultaneously estimate their position in an unknown environment and simultaneously create a map of the environment. This challenge is also referred to as the simultaneous localization and mapping (SLAM) problem. While filter-based approaches were the most common solution for this problem before 2010, graph SLAM is now the most popular and efficient approach [1, 2]. In this approach, a robot's landmarks and poses are represented by a graph, which allows the SLAM problem to be solved via nonlinear optimization techniques.

The SLAM problem refers to the difficulty of locating and mapping a mobile robot in an unknown environment with its simultaneous positioning relative to this map [3]. When no other navigation capabilities, such as GNSS, are available, the SLAM problem becomes more important. While already solving the problem in simple applications, SLAM algorithms can be pushed to their limits by challenging dynamic robot motions or highly dynamic environments [4]. To obtain a map, sensors must be used to detect the structure of the environment. A variety of possible sensor types are available for this purpose.

Finally, by determining the position of the environmental features, it is possible to obtain a representation of the robot's environment and thus a map that can be used in various ways, such as localization. The basic problem within SLAM is to estimate the trajectory of the robot as well as the position of all environmental features without knowing the true position of the features or the robot itself [3, 5]. LiDAR sensors, in particular, play a key role in sensing the environment of mobile systems [6, 7].

II. Range Sensing Basics

Range sensors are devices that capture the 3-D structure of surrounding objects from the sensor's perspective. They typically measure the distance to the nearest surfaces - that part of the scene is "visible" from the sensor. There is no full three-dimensional observation of all sides of the scene, and so in the field of sensory research is increasingly talking about two-and-a-half dimensional (2.5-D) data. (2.5-D).

The range data is a two-dimensional (2.5-D) or three-dimensional representation of the environment around the robot. The three-dimensional aspect arises because the coordinates (X, Y, Z) of one or more points in the scene are measured. But usually, only the fronts of objects are observed - that part of the scene that is visible from the robot. Typically, we do not have a full three-dimensional observation of all scene sides. Hence the term "2.5-D."

Two basic range measurement technologies exist *triangulation* and *time-of-flight* measurement, and there are multiple variations exist of each.

A. Triangulation

Triangulation sensors measure depth by determining the angle formed by rays from a world point to sensors located at a distance b. This so-called baseline of length b separates the sensors, assuming for simplicity that one of the rays forms a right angle with the baseline. Angle θ of the other sensor ray is then related to the depth Z perpendicular to the baseline through the relation:

$$\tan\theta = \frac{Z}{b}. \tag{1}$$

An image sensor measures the angle θ by an offset in the image plane relative to the primary beam. This shift x is denoted as misalignment. If the image plane is assumed to be parallel to the baseline, then $\tan\theta = f/x$, and the basic equation of triangulation depth sensors is obtained

$$Z = \frac{fb}{x}. \quad (2)$$

B. *Time of Flight*

The method used in time-of-flight (TOF) sensors is like that of a radar: it measures the time it takes light to reach an object and return. Since light travels at about 0.3 m per nanosecond, a very accurate TOF measurement is needed.

As they measure time-of-flight, these sensors can theoretically have constant accuracy irrespective of the distance to the object - unlike triangulation sensors, whose accuracy decreases with the square of the distance to the object. However, TOF sensors cannot achieve the high accuracy of triangulation sensors for closely spaced objects, and so they are not typically used in close-range applications.

1) Direct Time of Flight

In direct TOF sensors, travel time is measured with a high-speed stopwatch. Direct TOF laser range sensors are also called LiDAR or LaDAR (laser radar sensors). The travel time multiplied by the speed of light (in a given medium - space, air, or water and corrected for the density and temperature of the medium) gives the distance

$$2d = ct, \quad (3)$$

where d is the distance to the object, c is the light speed, and t is the travel time. The error in the measurement of time t results in a proportional error in the distance. In practice, an attempt is made to measure the peak of the output pulse, which has a finite range; weak reflections from distant objects make this peak difficult to measure, and therefore the error tends to increase with the distance. Multiple readings averaging can reduce the random error in these measurements [8].

The simplest TOF sensors use only a single beam, so range measurements are obtained from only one point on the surface. Robotics applications typically need significantly more information. To obtain this information, the laser beam is moved across the stage. Typically, the beam is moved using a set of mirrors [8].

Typical ground based TOF sensors suitable for robotics applications have a range of $10 - 100\ m$ and an accuracy of $5 - 10\ mm$. The volume of the scanned scene depends on the speed of the moving mirrors and the pulse rate, with typical values $1000 - 25000$ points per second.

The scanning multibeam LiDARs can increase the amount of information available. Companies such as Velodyne and Oster produce devices with 16, 32, 64, and 128 vertically aligned beams that acquire point data at up to 15 $scans/s$ (1.3 $MPixel/s$), with a full 360-inch horizontal scan and 27-inch vertical FOV from the laser array. The laser pulse length is 5 ns and the depth accuracy is approximately 2 cm. These devices are frequently used in environmental reconstruction, autonomous driving, and obstacle avoidance.

Flash LIDAR

Flash LIDARs have a two-dimensional detector array, unlike scanning devices. Instead of one or several laser beams, the light source pulse is shaped to cover a large area. All pixels start their timers when the pulse is initiated and measure the time it takes to receive the backscattered light. Typically, a few dozen samples are taken and averaged to reduce noise in the measurements - the amount of energy received is quite small since the laser is not focused on a beam. The pixels on the detector array are quite large because of the timing electronics; a typical ASC device has $128x128$ pixels and can capture data at up to 60 Hz. These devices are expensive and therefore not used in consumer applications [8, 9].

2) Indirect Time of Flight Sensors

In indirect TOF sensors, the distance is measured and transit times are determined from certain properties of the propagating beam. The two most important methods are based on modulation and phase differences and signal intensity [8].

III. Implementation

A. *Hardware*

1) LiDAR OS1

Fig. 1 shows the OS1 LiDAR from the company Ouster, which was used for this work. With the help of an interface box, the LiDAR sensor can be connected to a computer via a LAN cable. For the operation of the sensor, a 24 V power supply is also required.

The Ouster LiDAR is suitable for distances between 0.3 m to 100 m and has a vertical field of view of 45° (±22.5°). It has a vertical resolution of 128 lines and a configurable horizontal resolution of 512, 1024, or 2048 lines. Depending on the resolution, the LiDAR sensor can scan its environment at 10 Hz or 20 Hz. Thus, at a resolution of 128x2048 and the frequency of 10 Hz, up to 2621440 points can be captured by the sensor within one second, corresponding to a data rate of up to 254 Mb/s [10].

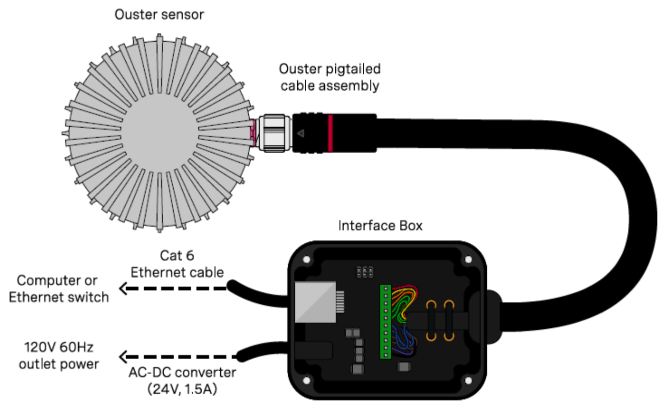

Ouster sensor
Ouster pigtailed cable assembly
Interface Box
Cat 6 Ethernet cable
Computer or Ethernet switch
120V 60Hz outlet power
AC-DC converter (24V, 1.5A)

Fig. 1. Ouster OS1 with 128 lines [11].

For later outdoor applications, the raster size is also of interest. The determination of the raster size with which the environment is scanned, as a function of the distance, can be determined using trigonometric relationships. This relationship is shown in Fig. 2.

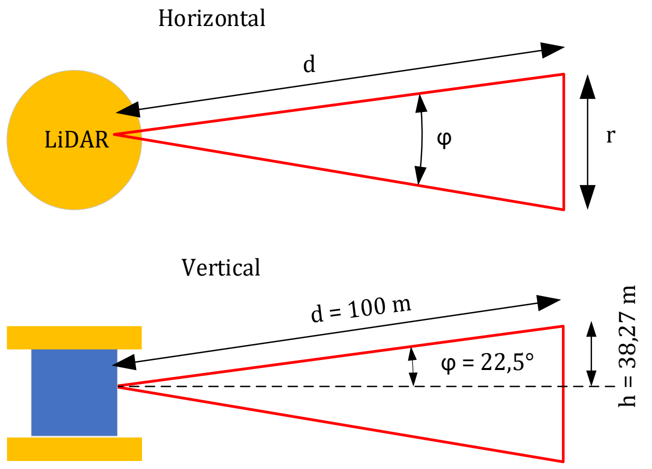


Fig. 2. Up: Horizontal distance between two laser beams as a function of distance; Down: Vertical field of view as a function of distance (100 m).

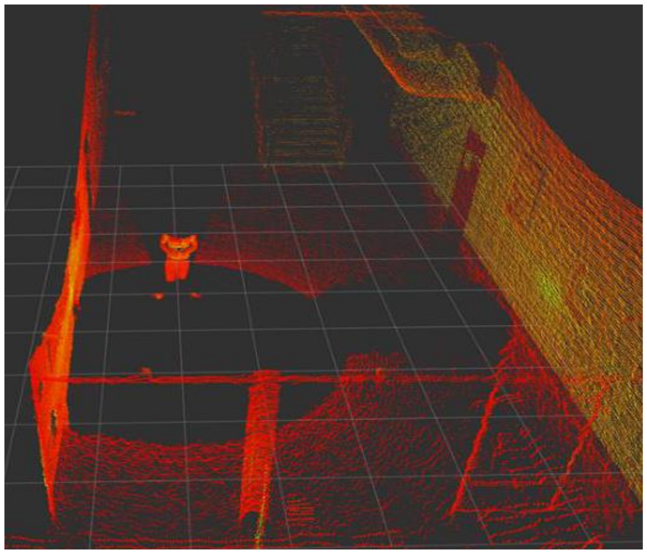
On the one hand, the maximum distance of 100 *m* is considered, which is particularly interesting for outdoor applications. On the other hand, a distance of 10 *m* is considered for the indoor area. The raster values, depending on the resolution and the distance, are listed in Table 1.

Table 1. Distance of the measuring points depends on the distance [12].

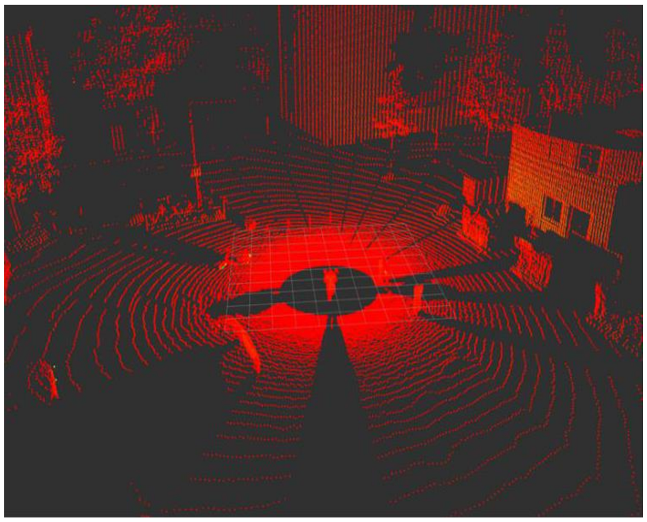
Alignment	Resolution, points	Field of view, °	Angle betw. 2 beams, °	Distance, m	Raster-size, cm
Vertical	128	45	0.35	100	61.36
				10	6.14
Horizontal	512	360	0.70	100	122.72
				10	12.27
	1028	360	0.35	100	61.12
				10	6.11
	2048	360	0.18	100	30.68
				10	3.07

For distances in the range of 10 *m*, a vertical grid of approx. 6 *cm* and a horizontal grid between 3 *cm* and 12 *cm* is obtained. In the outdoor area, a vertical grid of approx. 61 *cm* and a horizontal grid between 30 *cm* and 120 *cm* are obtained.

For indoor applications, where distances are smaller and there are many flat surfaces such as walls and floors, a horizontal resolution of 512 measurement points is sufficient. In the outdoor area, where there are considerably larger distances and increasingly complex structures such as trees, a horizontal resolution of 2048 measuring points is advantageous. This can be seen, for example, in the following images of the laser scanner in the indoor and outdoor areas (Fig. 3).



a)



b)

Fig. 3. Laser scanning in a) indoor and b) outdoor areas with the Ouster LiDAR OS1.

2) Husky A200

The UGV (Unmanned Ground Vehicle) "Husky A200" from Clearpath Robotics, which can be seen in Fig. 4, is used for this work.

The robot platform, which is equipped with an all-wheel-drive, can be remotely controlled with the aid of a controller. In this chapter, the structural design of the robot platform is described. First, the hardware structure is considered. This is followed by a brief general introduction to ROS (Robot Operating System) and the software structure.

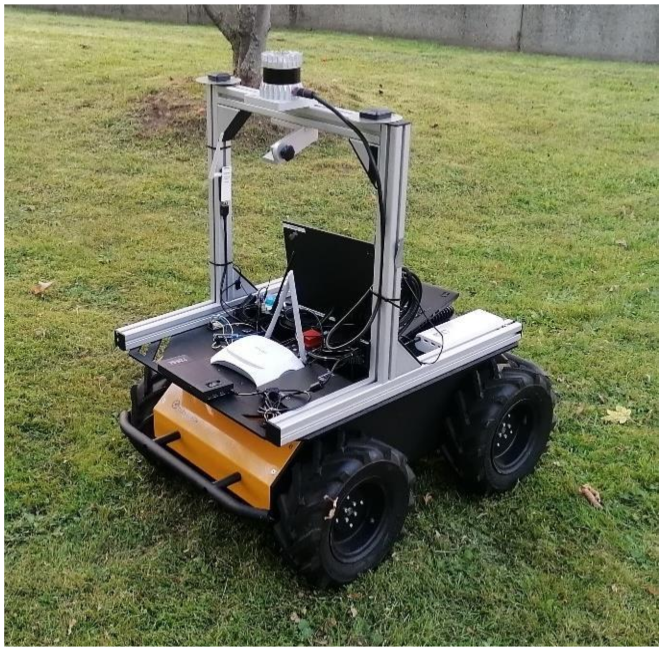


Fig. 4. Laser scanning in indoor and outdoor areas with the Ouster LiDAR OS1 [12].

A structural diagram of the hardware setup is shown in Fig. 5. Inside the Husky is the main computer, which contains the basic system for the Husky. When the rover is switched on, the main computer and thus all necessary processes are started automatically. The MCU (Motor Control Unit) is also connected to the Husky computer via a serial RS232 interface, which controls the motors of the robot platform and receives the odometry data [13].

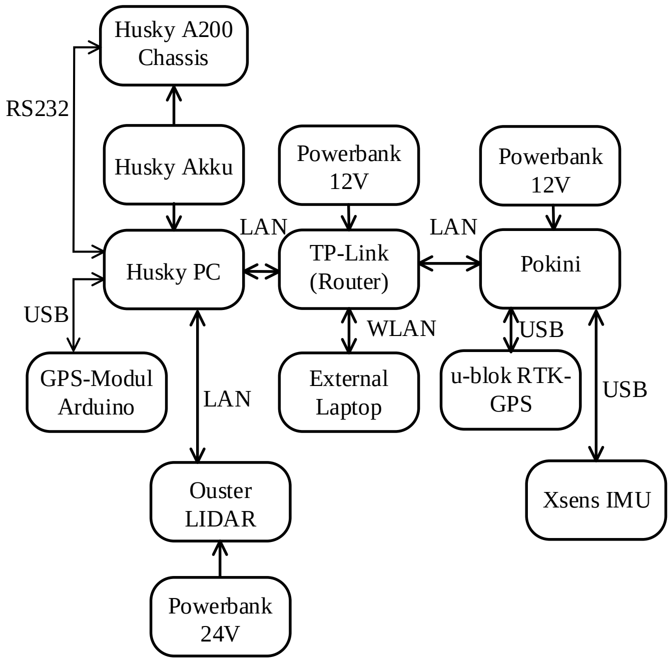


Fig. 5. The hardware structure of the robot platform (based on [13]).

In addition to the main computer, there is a second computer (Pokini) that is connected to the main computer via a router. The two computers are each connected to different sensors and have the necessary software packages accordingly. A GPS (Global Positioning System) module is connected to the Husky as well as the 128 lines LiDAR OS1 from Ouster. The Pokini is equipped with an IMU (Inertial Measurement Unit) from X-sense for inertial data acquisition and an RTK-GPS (Real-Time Kinematic-GPS) from u-blox. This will later be used as a reference to evaluate the accuracy of the trajectory estimation. The two computers can be accessed and communicated with via the WLAN router using an SSH connection.

B. Software

ROS is used as the framework for exchanging messages and controlling the Husky. In the following, the functionality of ROS is described first and then the structure of the ROS network of this robot platform is described.

1) Robot Operating System (ROS)

ROS (Robot Operating System) is an open-source platform for programming robot systems. It is a meta-operating system that provides various tools for simulation and visualization as well as libraries. The libraries mainly include hardware abstractions as well as device drivers for sensors and actuators [14].

ROS is a peer-to-peer network, which means that all participants have equal rights and that services can be offered and used. The individual participants can be distributed over several computers, whereby they are loosely coupled by the ROS communication infrastructure. To organize communication, there is exactly one master in each ROS network that manages communication and with which each node must be registered, as illustrated in Fig. 6.

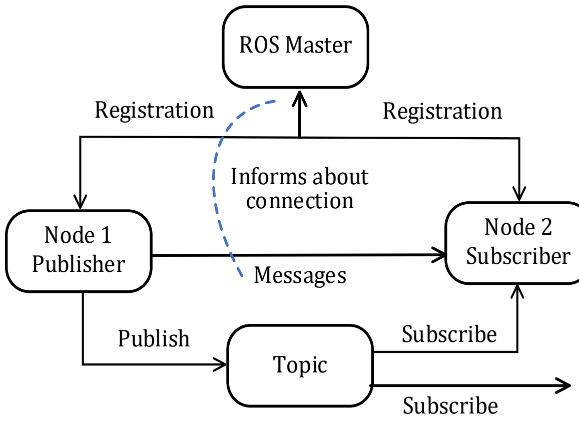


Fig. 6. Structural organization of ROS [14].

ROS nodes represent programs executable via a terminal, which can be compiled, executed, and managed individually. The organization of the Nodes takes place in packages, which contain the source code, Launch files, and configuration files [14, 15] .

The nodes communicate with each other via so-called topics. Each node can publish different topics or subscribe to topics that are of interest to it. For example, sensor data can be shared via a topic and subscribed to by the nodes that require this sensor information.

2) *ROS structure Husky*

When the Husky is started, a ROS core, which represents the master, is automatically started on the Husky computer. All necessary ROS packages that are required for the operation of the Husky are executed. If the corresponding sensors are to be started or used, they must be started manually, for example via an SSH connection, on the respective computer on which the corresponding software packages are installed. For example, the node of the Ouster LiDAR sensor is started on the Husky computer. The exact description of which packages can be started on which computer and how is given in the GitLab repository. Finally, the hardware setup, as described in Fig. 5, allows for subscribing or publishing the desired topics from any computer, enabling easy cross-network data exchange.

IV. EXPERIMENTAL RESULTS WITH HDL_GRAPH_SLAM

A. *The parking lot of the University of Offenburg*

The first outdoor test was recorded in the parking lot of Offenburg University. On the day of the test, the parking lot was sparsely filled with cars, as can be seen in Fig. 7.

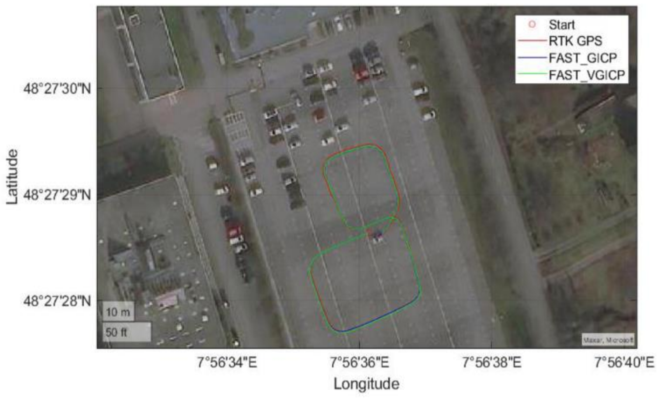


Fig. 7. Trajectories of the different scan matching methods (FAST_GICP, FAST_VGICP, and RTK GPS as ground truth) in the parking lot of the Offenburg University.

The parking lot is almost flat and very wide so that only ground surfaces are detected near the Husky robot, and trees and buildings can be detected by the LiDAR sensor from approx. 40 m. The LiDAR sensor is also able to detect trees and buildings. As a ground truth (reference), the trajectory was also determined using an RTK GPS. For this purpose, the base station was calibrated at the position of the car at the starting point. The RTK GPS can achieve the accuracy of 2 cm when the base station is calibrated. The total distance of the trajectory is 180 m with a duration of 223 s.

For verification, different scan matching methods for odometry determination are tested again. As in the indoor area, FAST_GICP and FAST_VGICP were shown to be the most suitable for this application. While the hdl_graph_slam with NDT_OMP deviates significantly from the ground truth up to the first loop closure, the deviation of the hdl_graph_slam with the other two algorithms is a maximum of 20 cm (Fig. 8).

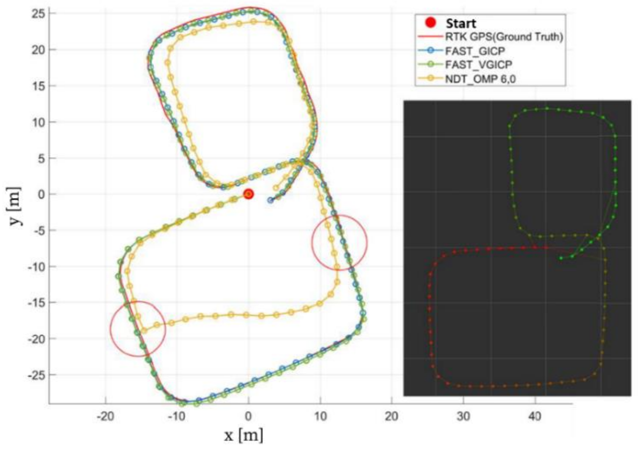


Fig. 8. Left: Trajectories of the different scan matching methods (FAST_GICP, FAST_VGICP, NDT_OMP, and RTK GPS as ground truth) in the parking lot of the University of Offenburg. Right: Trajectory with four loop closures found.

After the first Loop Closure, the deviation is reduced to 2 cm with all scan matching algorithms. In the second loop, the deviation with NDT_OMP is up to 3 m, but decreases again to a maximum of 1 m at the second loop closure. With FAST_GICP and FAST_VGICP, the deviation is 50 cm at the most and decreases again at the Loop Closure at the end to 20 cm at the most . The loop closures found by hdl_graph_slam can be seen on the right-hand side in Fig. 8.

B. *Check of the altitude course*

In addition to the trajectory, the altitude course is also checked. The RTK GPS cannot be used as a reference, since its elevation values vary by several meters (Fig. 9, b) and this is a flat parking lot. The values of the hdl_graph_slam, however, only vary by ±5 cm.

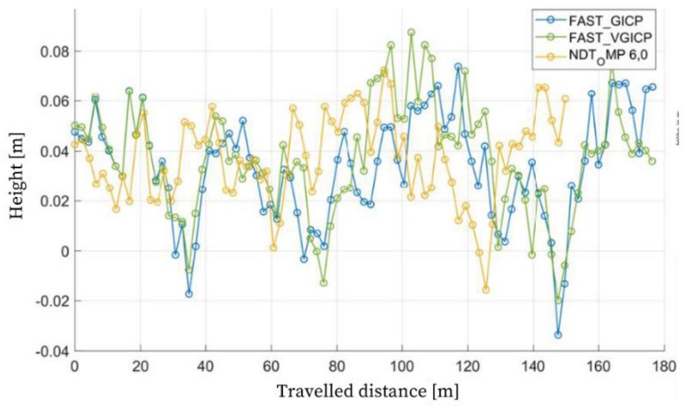


Fig. 9. Elevation course of the trajectory of the different scan matching methods (FAST_GICP, FAST_VGICP, and NDT_OMP) on the parking lot of the University of Offenburg.

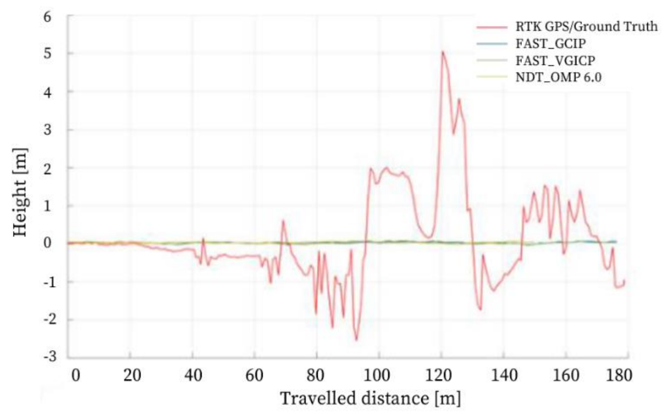


Fig. 10. Altitude course of the RTK GPS (red).

V. Conclusion

This paper presents a system for the implementation and evaluation of complex localization and mapping procedures. The considered methods are suitable for reliable localization in complex scenarios, such as logistics applications in Industry 4.0, exploration and sensing in field robotics, or applications in service robotics.

Currently, a major problem is the precise positioning of mobile systems, which is a prerequisite for any autonomous behavior in industrial environments and field robotics. The paper describes the setup of an experimental platform and its use for the evaluation of SLAM algorithms

With each scan of a LiDAR sensor, a point cloud with measurement points is obtained, as shown in the previous chapter. To be able to use these point clouds for mapping or localization in a map, the challenge is to "match" the current scan with an already known scan of the surrounding area. The process of matching two scans is also called registration. There are different scanmatching algorithms for registration.

Among the most common are the Iterative Normal Distribution Transform (NDT), Closest Point (ICP), and Voxelized Generalized Iterative Closest Point (VGICP) algorithms. In the next stages of the study, different algorithms will be investigated and compared in indoor and outdoor spaces.

Acknowledgments

This research is supported by the Bulgarian National Science Fund in the scope of the project “Exploration the application of statistics and machine learning in electronics” under contract number КП-06-Н42/1.

References

[1] M. Ivanova, P. Petkova and P. Petkov, "Machine Learning and Fuzzy Logic in Electronics: Applying Intelligence in Practice," *Electronics,* vol. 10, no. (22):2878, 2021.

[2] S. Hensel, M. B. Marinov, C. Kehret and M. Stefanova-Pavlova, "Experimental Set-up for Evaluation of Algorithms for Simultaneous Localization and Mapping. In: Yilmaz M., Niemann J., Clarke P., Messnarz R. (eds.) Systems, Software and Services Proces," *Systems, Software and Services Process Improvement. EuroSPI 2020. Communications in Computer and Information Science,* vol. 1251, pp. 433-444, 2020.

[3] H. Durrant-Whyte and T. Bailey, "Simultaneous Localisation and Mapping (SLAM): Part I The Essential Algorithms," *Robotics and Automation Magazine,* vol. 2, p. 1–9, 2006.

[4] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid and J. Leonard, "Past, Present, and Future of Simultaneous Localization and Mapping: Towards the Robust-Perception Age," *IEEE Transactions on Robotics,* vol. 32, no. 6, 2016.

[5] J. Zhang and S. Singh, "LOAM: Lidar Odometry and Mapping in Real-time," in *Robotics: Science and Systems Conference,* Berkeley, 2014.

[6] H. Weber, "Funktionsweise und Varianten von LiDAR-Sensoren," [Online]. Available: https://cdn.sick.com/media/docs/5/25/425/whitepaper_lidar_de_im00794 25.pdf . [Accessed Oct. 2021].

[7] I. Maksymova, C. Steger and N. Druml, "Review of LiDAR Sensor Data Acquisition and Compression for Automotive Applications," *Proceedings,* Vols. 2, 852, 2018.

[8] B. Siciliano and O. Khatib, Eds., Springer Handbook of Robotics, Springer, 2016.

[9] W. Hess, D. Kohler, H. Rapp and D. Andor, "Real-Time Loop Closure in 2D LIDAR SLAM," in *IEEE International Conference on Robotics and Automation (ICRA),* 2016.

[10] Ouster, "Ouster OS1: Mid-Range High-Resolution Imaging Lidar," [Online]. Available: https://ouster.com/products/os1-lidar-sensor/. [Accessed 20 Oct. 2021].

[11] Ouster, "OS1 Hardware User Manual," [Online]. Available: https://levelfivesupplies.com/wp-content/uploads/2019/03/OS-1-User-Guide-Hardware.pdf. [Accessed 20 Sept. 2021].

[12] M. Obert, "Inbetriebnahme und Evaluierung des hdl_graph_slam mit einem 128 Zeilen Ouster LiDAR -Sensor auf der Husky Roboterplattform von Clearpath," Hochschule Offenburg, Offenburg, 2021.

[13] C. Kupitz, "Inbetriebnahme und Verifizierung eines Kalman Filter zur Lagebestimmung des Clearpath Robotics Husky A200," 2021.

[14] E. Jelavic, "ETH Zürich: Programming for Robotics, Introduction to ROS," [Online]. Available: https://rsl.ethz.ch/education-students/lectures/ros.html. [Accessed Feb. 2022].

[15] Blasdel et al., "About ROS.Version: 2020," [Online]. Available: https://www.ros.org/about-ros/.