19. AALE-Konferenz
Luxemburg, 08.–10. März 2023

Low-Cost Automation: An Open Source Laser-Triangulation Sensor based on ROS2

Moritz Schauer[1], Tristan Elias Wolfram[1], Jakob Czekansky[1], Diethelm Bienhaus[1]

Abstract

This paper presents the development of an open-source laser triangulation sensor system for automation tasks. The prototype system is based on the Robot Operating System 2 (ROS2) and aims to provide a low-cost alternative to commercially available sensors while achieving comparable accuracy. The proposed system aims to facilitate the adoption of automation in smaller applications and make it more accessible, particularly in academic contexts. With the system, we achieve an average depth estimation deviation of approximately $200\,\mu\mathrm{m}$.

Keywords

Sensors, Low-Cost Automation, ROS 2, Robotics, 3D Perception

1 Introduction

In a more and more automated industrial manufacturing environment, robots are playing an ever-increasing role. Especially for small and medium-sized companies, robots are becoming increasingly attractive, primarily because the hardware is becoming more affordable. However, automation does not stop with acquiring a robot and its programming. Additional sensors are needed for a fully autonomous system that can react to changing surroundings, especially in rapidly changing production environments. Imaging sensors play a significant role here, regardless of whether the data is 2D or 3D. Nevertheless, 3D sensors can provide more information with additional depth data, through which a better statement about the environment can be made. For constructing such 3D sensors, one has the choice of different techniques. The most common are Time-of-Flight (ToF) [1], active or passive stereo [2], and structured light [3]. Thereby, laser triangulation is best associated with the latter. Laser triangulation is a precise, non-contact measurement technique that is used in a variety of industrial applications, such as inspection and quality control [4], robotic guidance [5], and 3D scanning [6]. Even though these sensors are now well established in the industry, the commercially available systems are very cost-intensive. This represents a financial hurdle for smaller applications, especially in an academic context. For this reason, in this paper, we propose an open-source laser triangulation sensor system based on the Robot Operating System 2 (ROS2). In the design of the prototype system, we put more weight on the low-cost concept to keep the system's overall cost low. Moreover, in addition to the low-cost approach, we ensured that the sensor has comparable accuracy to sensors used in the industry. Since the sensor is open hardware, it can easily be adapted to changing requirements. For example, shorter working distances can be realized. This can be particularly interesting if both the depth data and the texture of surfaces are to be recorded with a high resolution and accuracy, for example, in dendrochronology [7].

---

[1] Institut für Technik und Informatik, Technische Hochschule Mittelhessen, Gießen

2 Concept

The principle of laser triangulation is based an the projection of a 3D point, represented as P_W, onto the image plane using a perspective transformation. This projection creates a corresponding pixel, represented as p_I. The perspective transformation used for this projection is known as the pinhole camera model. The pinhole camera model provides a distortion-free projective transformation, as shown in equation (1).

$$s\, p_I = A\, [R \mid T]\, P_W \tag{1}$$

The camera intrinsic matrix is represented as A, while R and T are the rotation and translation that describe the transformation from the world to the camera coordinate system (or camera frame). The scaling factor s is an arbitrary element of the projective transformation and is not included in the camera model. However, it can be calculated during extrinsic calibration of the camera.

Instrinsic Calibration

But first the intrinsic parameters of the camera have to be determined. These are the focal lengths f_x and f_y, the principal point (c_x, c_y), and the radial and tangential distortion coefficients. For the calibration process a calibration target is needed. Normally, a simple chessboard pattern would suffice here. However, for the later extrinsic calibration we need two patterns orthogonal to each other. Since the two chessboard patterns would be difficult or impossible to distinguish, we used so-called ChArUco borads for the calibration process. These are chessboard patterns which have AruCo markers in the white squares. By using different markers for the two chessboards, they can be precisely distinguished. To estimate the intrinsic parameters we use the built-in methods of OpenCV, which rely an Zhang's method [8].

Laser Line Extraction

Single points on the laser line must be detected for the extrinsic calibration and the calculation of the depth data after the calibration. In order to obtain accurate subpixel points, a few pre-processing steps are necessary. In the first step, a difference image of two images is created. Therefore, we subtract an image where the laser is off from one where it is switched on. The difference image is filtered in the next step with a Gaussian filter of size 5. The filtering reduces noise in the image. Then the laser line is searched line by line in the image. First, a global threshold is calculated using the Otsu thresholding method [9]. With this threshold value, we check line by line if there is a pixel above this value. If no pixel is detected, the line is not considered further. We determine the center of the laser line with subpixel accuracy in each line where at least one pixel is considered valid. First, the pixel with the highest intensity I_{max} is determined a position x. Then a quadratic function is laid through the pixel and its two neighbors using (2).

$$\begin{pmatrix} x^2 & x & 1 \\ (x+1)^2 & (x+1) & 1 \\ (x-1)^2 & (x-1) & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} I_x \\ I_{x+1} \\ I_{x-1} \end{pmatrix} \tag{2}$$

Since we are not searching globally in the line but only looking for a deviation of at most one pixel around I_{max}, we can set $x = 0$ and make the following simplification:

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} I_x \\ I_{x+1} \\ I_{x-1} \end{pmatrix} \tag{3}$$

Now we can solve the equation system and get the coefficients of the quadratic function. To find the maximum and, thus, the center of the laser line, the zero of the derivative is calculated. For this applies:

$$\begin{aligned} 2a \cdot x + b &= 0 \\ x &= \frac{-b}{2a} \end{aligned} \tag{4}$$

By substituting the determined values from (3) we get a number in the interval $[-1, 1]$, which indicates the deviation of x around the pixel I_{max}.

Extrinsic Calibration

The goal of the extrinsic calibration is to calculate the laser plane equation (LPE). During point cloud generation, the LPE can convert the calculated subpixels into threedimensional points. To calibrate the LPE, we need a calibration pattern. For this, we use two ChArUco-boards arranged at a right angle to each other. With this configuration, we get two laser lines that have a unique orientation and translation to each other. One can see the setup of the calibration boards with the laser line in fig. 4.

The calibration of the LPE can then be divided into the following steps: (a) Detection of both ChArUco-boards, (b) calculation of the orientation of both boards, (c) definition of two ROIs for the laser line, (d) extraction of both laser lines, (e) transformation of both laser lines into the same coordinate system, (f) fitting of the LPE.

We calculate an ROI based on the ArUco-marker detection to extract the two separate laser lines. This gives us a rectangle for each board, which we can then use to trim the image. We can use the pinhole camera equation to transform the laser line points into a shared coordinate system (1). During the calibration process, we can now calculate the missing scaling factor s. For this we first rearange (1) as following:

$$P_W = s\,\underbrace{[R]^{-1}\,[A]^{-1}\,p_I}_{\vec{a}} - \underbrace{[R]^{-1}\,t}_{\vec{b}} \tag{5}$$

In the next step, we lay our world coordinate system in the origin of the ChArUco-boards. Since we can assume that all points of the respective laser line are located in the XY plane of the respective ChArUco board, we can set $P_{W_z} = 0$ in (5). From this, follows:

$$\begin{aligned} P_{W_z} &= s\,a_z - b_z \\ 0 &= s\,a_z - b_z \\ s &= \frac{b_z}{a_z} \end{aligned} \tag{6}$$

Now we can extract the three-dimensional coordinates of all points of both laser lines. The second to last step transforms all points into a standard coordinate system. Finally, a plane is fitted through all given laser points using the SVD method. The resulting LPE is later used to calculate the depth data during each scan.

3 Architecture and Design

In this section, we will briefly elaborate on different aspects of our architecture and on design decisions. We will start by outlining the hardware setup in the first subsection, followed by a detailed description of the software architecture. We will conclude our description by pointing out our modularity approach and the usage of the developed ROS nodes.

3.1 Hardware Setup

We built a prototype for a proof of concept of our architecture. The initial focus was on the availability of parts and simple control. The low-cost approach was already partially pursued but can also be

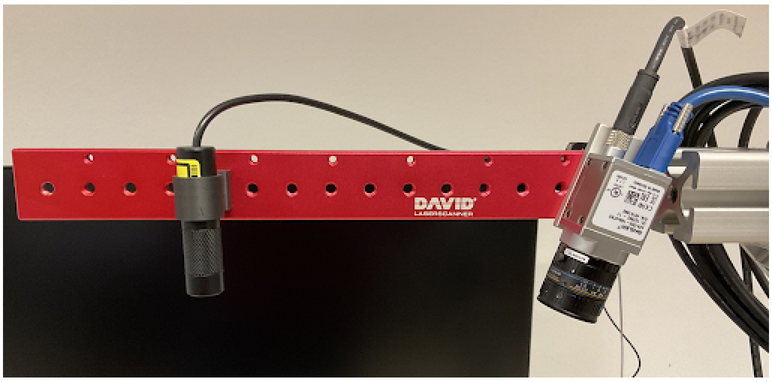

Figure 1: Prototype setup consisting of a line laser and an industrial camera.

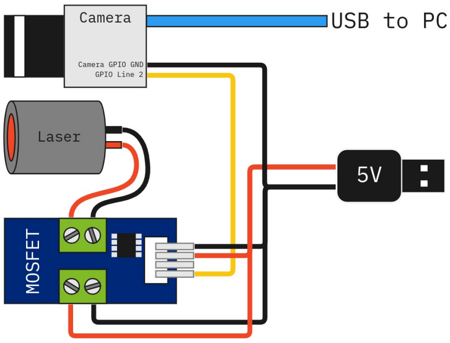

Figure 2: Wiring diagram of the prototype.

expanded. The laser used was a modul[1] from Picotronic. This is a 650 nm laser module with 5 mW power and an adjustable focus. It can be equipt with Diffractive Optical Elements (DOE) to generate different lighting patterns. For our purpose we add a solid line DOE wich produces a 1 mm thick line and has a FOV of 45.5°. To acquire the images, we used an industrial camera[2] from Basler. The advantage of the industrial camera for our prototype is that it has I/O ports. So no extra I/O controller must be connected to the computer to control the laser.

The basis of the prototype is an aluminum profile, which has a grid of ¼" - 20 UNC threads. The distance between the holes is 12 mm. Both the camera and the laser are attached to this profile. The laser is oriented vertically downwards so that the projected line is orthogonal to the profile. The camera is oriented at an angle of 60° to the laser. The distance between the laser and the camera is 140 mm. One can see the mounted prototype in fig. 1. As we use a higher working distance than typical industrial sensors we did not incorporate optics according to the Scheimpflug rule like in Chen et al. [10]

Since the sensor is designed to record colorized depth data, it must be possible to switch the laser on and off. For this, we use one of the optoisolated outputs of the camera. This camera output is then connected to a MOSFET, which controls the laser's power supply. The wiring can be seen in fig. 2.

3.2 Software Implementation

We implemented the software using the Python API of ROS 2. Thereby, the software implementation for the scanner consits of two main nodes that provide the necessary functionality through their services. In order to use the scanner, a client node must be employed to call theses services. The concept of nodes and their respective services can be observed in fig. 3.

The ROS node `camera_node` represents the physical camera and is responsible for obtaining the images. Therefore the node offers different services that can trigger different types of image acquisition. In general, a distinction can be made between calibration and regular scanning operation. During the calibration process, the camera can send multiple images of the chessboard pattern for intrinsic calibration through the `cam_calib_imgs` topic. The second topic the `camera_node` publishes, is `img_pair`. Here the image pair consists of one image where the laser is switched on and one where it is off. As the camera is directly connected to the laser (as shown in fig. 2), it can toggle it by itself during image acquisition. Additionally, a variable is set to determine if the images are being used for calibration or the generation of point clouds.

The second ROS node is the `surface_scanner_node`. It is responsible for generating the point clouds for a given image pair. But in order to produce a point cloud, the scanner needs to be calibrated. Here, it is responsible for both intrinsic and extrinsic calibration. It also stores the calibration data like the

---

[1] LD650-55(12x32)45-F300

[2] a2A1920 - 160ucPRO

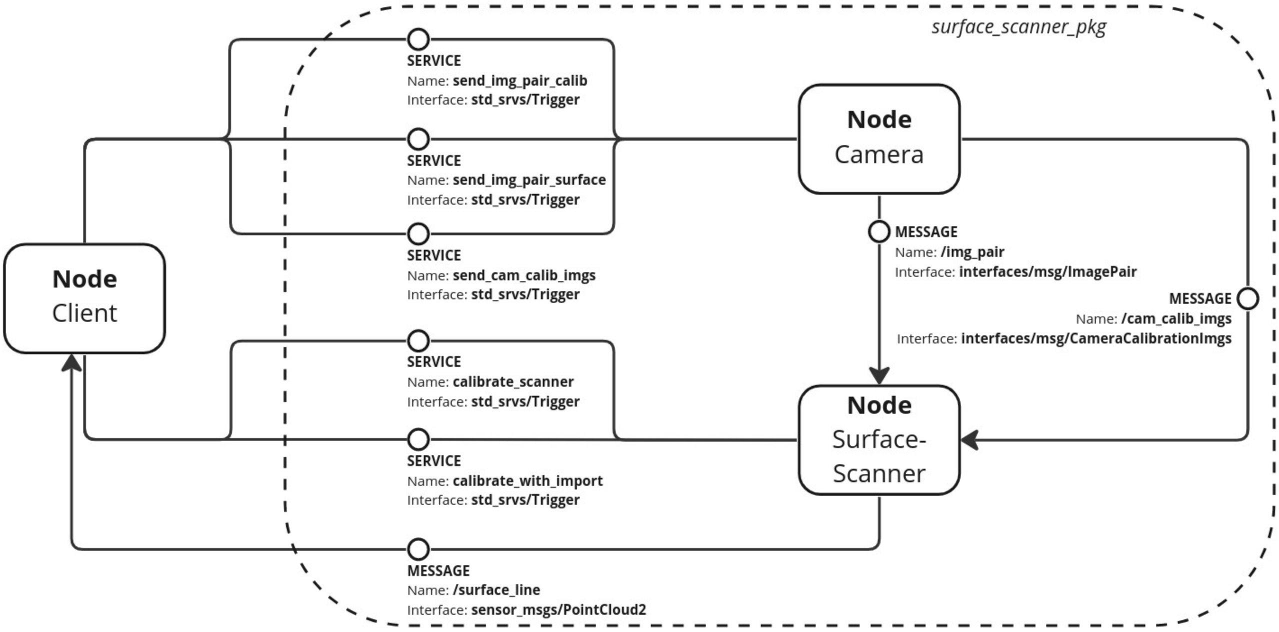
surface_scanner_pkg
Node Client
Node Camera
Node Surface-Scanner
SERVICE
Name: send_img_pair_calib
Interface: std_srvs/Trigger
SERVICE
Name: send_img_pair_surface
Interface: std_srvs/Trigger
SERVICE
Name: send_cam_calib_imgs
Interface: std_srvs/Trigger
SERVICE
Name: calibrate_scanner
Interface: std_srvs/Trigger
SERVICE
Name: calibrate_with_import
Interface: std_srvs/Trigger
MESSAGE
Name: /surface_line
Interface: sensor_msgs/PointCloud2
MESSAGE
Name: /img_pair
Interface: interfaces/msg/ImagePair
MESSAGE
Name: /cam_calib_imgs
Interface: interfaces/msg/CameraCalibrationImgs

Figure 3: FMC block diagram showing all nodes, messages, and services of the ROS implementation.

laser plane equation, the intrinsic camera matrix, and the distortion coefficients. For the calibration process, the `surface_scanner_node` provides two services. The first service checks that the scanner has received all calibration images and will perform the calibration as a whole. For the second service, we implemented an option to calibrate the scanner by importing the camera data from intrinsic calibration. Since we use a camera with a fixed focus, the intrinsic camera parameters do not change. Therefore the intrinsic calibration has to be done only once. However, an extrinsic calibration must always occur when the scanner is re-initialized. The reason is that the prototype is not that rigid, and even a slight deviation in the geometrical relations would invalidate the extrinsic calibration. Finally, after the calibration is done, the scanner can be used to obtain depth data out of image pairs send by the camera. The workflow for that is described in fig. 5. A client node calls the service of the camera node responsible for obtaining an image pair for surface reconstruction. If an image pair is received and the scanner has been calibrated, a point cloud is automatically generated and published through the `surface_line` topic. The point clouds generated by the scanner can be received by any node capable of processing the `PointCloud2` message type, such as *rviz2*[3]. This generated point cloud is a cross-section of the scanned scene at the position of the laser line. The sensor or the scene must be moved to obtain a full surface reconstruction. The point clouds can then be merged in the correct spacing.

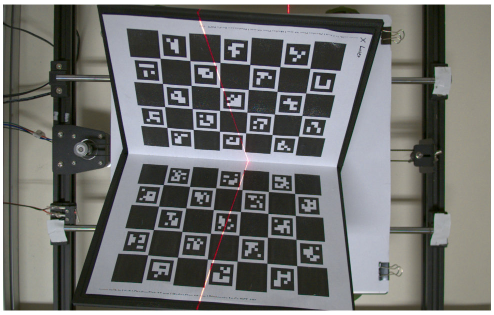

Figure 4: ChArUco-Board used during extrinsic calibration with the laser line projected on it.

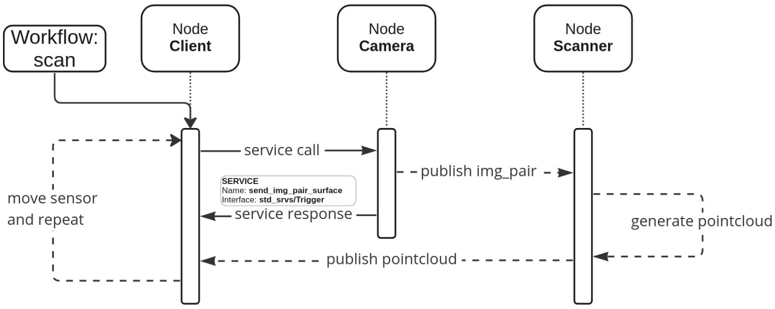

Figure 5: Workflow of the scanning process.

---

[3] https://github.com/ros2/rviz

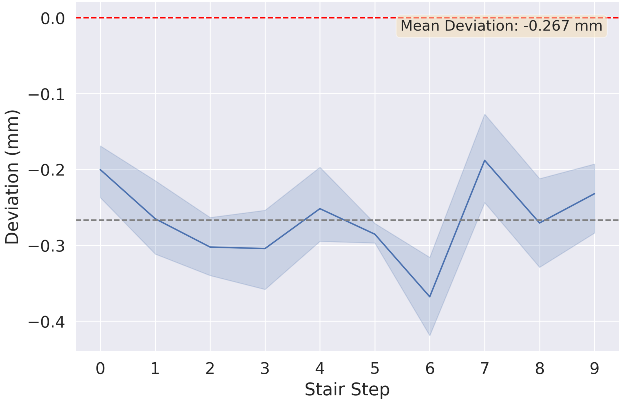

Figure 6: Plot of absolute deviation measured at a step size of 10 mm. The black line shows the average deviation.

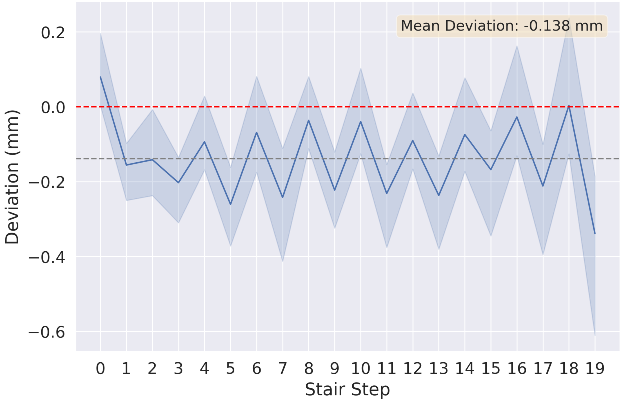

Figure 7: Plot of absolute deviation measured at a step size of 5 mm. The black line shows the average deviation.

4 Evaluation

In order to evaluate the sensor prototype, a test object was created. This test object is a 10 cm high staircase with steps spaced at 10 mm and 5 mm intervals. We manufactured the staircase using a high-precision 3D printer. The test object was placed under the sensor such that the generated laser line was centered on one of the rows of steps, and a working distance of 40 cm was used. An exemplary point cloud can be seen in figure 8. The scanner's accuracy was determined by calculating the distances between the individual steps. The first step was to segment the point cloud, creating different point clouds for the floor and the individual steps. Subsequently, a straight line was fitted into the point cloud of the floor with the help of singular value decomposition (SVD). We used the orientation of the floor line for the whole point cloud. Next, we need to calculate the lines for the individual stair steps. Therefore, we first calculated the mean value for each step. Afterward, lines with the same alignment as the floor line were laid through the mean values. In the last step, the distances of the individual lines to each other were calculated. In order to obtain better comparability, the total measurement was repeated six times for both series of steps. The results of the measurements are shown in figures 6 and 7, respectively. The dark blue line represents the mean value of the individual measurement points. The blue shaded area is the deviation around the mean value. In addition, the zero line is shown in red, and the total mean value is in black. In the measurement of the stairs with a step interval of 10 mm, the average step height deviation was $-267\,\mu\text{m}$. The average deviation for the staircase with a 5 mm interval was even lower, at $-138\,\mu\text{m}$. The maximum and minimum absolute deviation for the

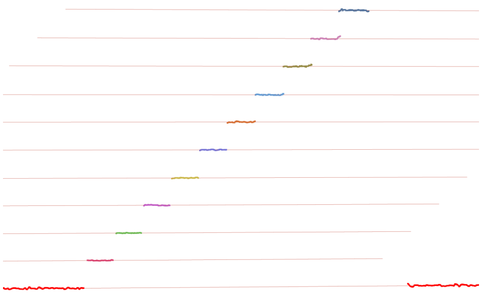

Figure 8: A plot of the generated point cloud of the staircase with a step interval of 10 mm. The fitted lines are also plotted.

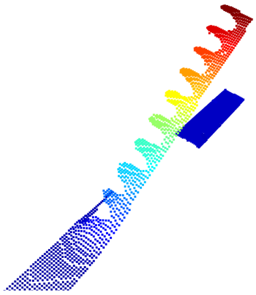

Figure 9: A scan of the printed staircase with the Intel RealSense d415. The steps with the 5 mm interval (on the right side) are not noticeable.

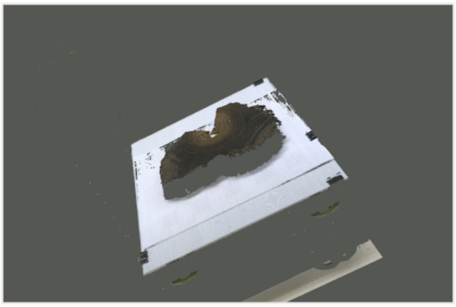

Figure 10: Scan of a tree slice by using a linear table.

stairs with a 10 mm interval was $483\,\mu\mathrm{m}$ and $71\,\mu\mathrm{m}$, respectively. Likewise, for the stairs with a 5 mm interval, the maximum and minimum absolute deviation were $1006\,\mu\mathrm{m}$ mm and $13\,\mu\mathrm{m}$, respectively.

In addition to the stair steps, we measured a line on a flat surface. Here we calculated the fitted line's Root Mean Squared (RMS) error. We repeated this measurement three times at different locations to suppress outliers in the measurements. Here the averaged RMS error is $102\,\mu\mathrm{m}$.

Additionally, we repeated the staircase measurement with a RealSense d415 from Intel. Here, the steps of the stairs with 5 mm intervals were net detectable at an operating range of 40 cm. The larger steps were detectable. Here the mean deviation was $149\,\mu\mathrm{m}$.

Besides the metrological measurements, we did a full-colored scan of a wooden slice. Therefore, we mounted the slice on a linear table. The table was then moved in 1 mm intervals under the scanner. Afterwards, the resulting scan lines were stitched to form one coherent point cloud. The results can be seen in fig. 10.

5 Conclusion and Future Work

We have shown how to implement a laser triangulation sensor setup with ROS2. Besides implementing the prototype, we also tested the sensor's accuracy by performing different measurements. Here we reached a mean deviation in the depth estimation between $-267\,\mu\mathrm{m}$ and $-138\,\mu\mathrm{m}$. We showed that these values are comparable to other 3D imaging systems like the Intel RealSense d415. We plan to increase the accuracy further by using more prices and robust calibration methods. In doing so, we will mainly focus on more accurate calibration targets.

Besides that, we plan to extend the prototype to a plug-and-play sensor. In this context, we want to emphasize the low-cost and open-source approach even more. Therefore, the sensor will be based on a single-board computer such as the Raspberry Pi. We want to provide a custom embedded operating system for the sensor to minimize the configuration. In addition, the industrial camera will be replaced by a cheaper board camera. Optionally, the camera could be adapted with appropriate optics to the Scheimpflug rule so that shorter working distances are possible with maintained accuracy. Furthermore, we plan to provide a 3d printed housing. The whole sensor is powered via Power over Ethernet (POE), a common standard for imaging sensors.

6 Acknowledgment

The writing of this work was enabled in part from within the context of the project "Development of a measurement methodology to enable a quick determination of the of wood species and origin on the basis of tree ring and color analysis". It is funded by the Federal Ministry for Economic Affairs and Climate Action (BMWi) as part of the "Central Innovation Program for small and medium-sized enterprises (SMEs)" under the funding code ZF4773401GR9.

7 Literature

[1] A. Kolb, E. Barth, R. Koch, and R. Larsen, "Time-of-Flight Sensors in Computer Graphics," in *Eurographics*, Munich, 2009.

[2] L. Keselman, J. I. Woodfill, A. Grunnet-Jepsen, and A. Bhowmik, "Intel(R) RealSense(TM) Stereoscopic Depth Cameras," in *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Honolulu, HI, USA: IEEE, Jul. 2017, pp. 1267–1276.

[3] S. Zhang, "High-speed 3D shape measurement with structured light methods: A review," *Optics and Lasers in Engineering*, vol. 106, pp. 119–131, Jul. 2018.

[4] V. I. Moreno-Oliva, E. Román-Hernández, E. Torres-Moreno, J. R. Dorrego-Portela, M. Avendaño-Alejo, M. Campos-García, and S. Sánchez-Sánchez, "Measurement of quality test of aerodynamic profiles in wind turbine blades using laser triangulation technique," *Energy Science & Engineering*, vol. 7, no. 5, pp. 2180–2192, 2019.

[5] J.-D. Sun, G.-Z. Cao, S.-D. Huang, K. Chen, and J.-J. Yang, “Welding seam detection and feature point extraction for robotic arc welding using laser-vision,” in *2016 13th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Aug. 2016, pp. 644–647.

[6] E. W. Y. So, M. Munaro, S. Michieletto, M. Antonello, and E. Menegatti, “Real-Time 3D Model Reconstruction with a Dual-Laser Triangulation System for Assembly Line Completeness Inspection,” in *Intelligent Autonomous Systems 12*, ser. Advances in Intelligent Systems and Computing, S. Lee, H. Cho, K.-J. Yoon, and J. Lee, Eds. Berlin, Heidelberg: Springer, 2013, pp. 707–716.

[7] M. Schauer, J. Czekansky, M. Kreutzer, and D. Bienhaus, “Robot-based image acquisition for dendrochronological analysis of curved wooden surfaces,” in *ISR Europe 2022; 54th International Symposium on Robotics*, Jun. 2022, pp. 1–6.

[8] Z. Zhang, “A flexible new technique for camera calibration,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[9] N. Otsu, “A Threshold Selection Method from Gray-Level Histograms,” *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, Jan. 1979.

[10] R. Chen, Y. Li, Y. Li, G. Xue, and X. Li, “Laser triangulation measurement system with Scheimpflug calibration based on the Monte Carlo optimization strategy,” *Optics Express*, vol. 30, no. 14, pp. 25 290–25 307, Jul. 2022.