Aalto-yliopisto
Aalto-universitetet
Aalto University

Master's Programme in Geoinformatics

Metrological characterization of a consumer grade flash LiDAR device

Antti Järvenpää

Aalto University

Master's Thesis
2021

Aalto-yliopisto
Aalto-universitetet
Aalto University

Author Antti Järvenpää

Title of thesis Metrological characterization of a consumer grade flash LiDAR device

Programme Master's Programme in Geoinformatics

Thesis supervisor Assistant Professor Matti Vaaja

Thesis advisor MSc. Heikki Kauhanen

Date 22.11.2021 **Number of pages** 51 + 34 **Language** English

Abstract

This work studied accuracy and characteristics of the depth sensing system built in Apple iPad Pro (2020) tablet device. The system was evaluated from the metrological point of view, and included inspection of artifacts affecting the use of it.
The work included theoretical review of the technology and error analysis. Though the work focuses on the accuracy of the flash LiDAR sensor, this part also included positioning methods of the system. Investigation of positioning methods was considered important, as it is directly linked to the total accuracy in practical applications.
In this work data was evaluated both visually, and statistically using measurement deviations. The experiments included inspection of illumination power distribution, accuracy dependency on both detection range and target material, significance of colour image feed in depth map creation and accuracy anomalies within a depth map.
During the work, the standard deviation of measurements was noted to be less than half a centimetre within the effective range, 0.2–5 m, of the device. However, the data quality was significantly affected by the enrichment of depth sensor observations with colour images. The system seemed to smoothen features and produce false depth values to sharp edges in a depth map. The system does not provide access to raw depth data, but only to this processed product more suitable for AR applications, where the system is intended to be used. As a result, the data processing solution is not optimal for measurement purposes.
Despite the shortages found in this work, the system has still potential for measurement purposes. Conventional methods providing the same data and information might be difficult, time consuming, or expensive compared to the tested technology, which makes it appealing. In addition, many of the found artifacts could also be filtered from the data with proper methods. In the future the applicability for data collection may improve also if the manufacturer decides to open access to unprocessed data.

Keywords Apple iPad Pro, LiDAR, time-of-flight, metrology, accuracy, precision

Aalto-yliopisto
Aalto-universitetet
Aalto University

Tekijä Antti Järvenpää

Työn nimi Kuluttajatason taulutietokoneeseen sulautetun syvyyssensorin käytettävyys mittaussovelluksissa

Koulutusohjelma Geoinformatiikan maisteriohjelma

Vastuuopettaja/valvoja Apulaisprofessori Matti Vaaja

Työn ohjaaja DI Heikki Kauhanen

Päivämäärä 22.11.2021 **Sivumäärä** 51 + 34 **Kieli** englanti

Tiivistelmä

Tässä työssä tarkasteltiin Apple iPad Pro (2020) -tablettitietokoneeseen sisäänrakennetun syvyyssensorin tuottaman datan käytettävyyttä mittaussovelluksissa. Työn kokeet keskittyivät mittaustarkkuuden arviointiin sekä mittaussovellusten kannalta keskeisten laitteen ominaisuuksien erittelyyn.
Työhön sisältyneessä kirjallisuuskatsauksessa selvitettiin sensorin teknologista taustaa sekä virhearvioinnin teoriaa. Vaikka työ muuten keskittyi sensorin ominaisuuksien selvittämiseen, kirjallisuuskatsauksessa selvitettiin myös laitteen paikannusmenetelmien tarkkuutta. Selvitys koettiin tärkeäksi paikannuksen tarkkuuden linkittyvän suoraan tiedonkeruun laatuun käytännön sovelluksissa.
Laitteen testauksessa aineiston laatua arvioitiin visuaalisesti, sekä tarkastelemalla aikasarjoista havaintojen hajontaa. Työssä tarkasteltiin laitteen valolähteen tehojakaumaa ympäristössä, tarkkuuden etäisyys- ja kohdemateriaaliriippuvuutta, tukevan värikuvan merkittävyyttä syvyyskuvien muodostuksessa sekä tarkkuuden vaihteluita yksittäisen syvyyskuvan sisällä.
Työssä havaittiin laitteen tuottamien syvyyshavaintojen olevan melko tarkkoja 0,2–5 m etäisyyksillä kohteesta. Huomioitavaa kuitenkin on, että laitevalmistajan kehittämällä syvyys- ja värikamerahavaintojen yhdistämisprosessilla on suuri vaikutus käyttäjälle saatavissa olevan aineiston laatuun, mikä ilmenee yksityiskohtien pehmenemisenä ja epätodellisina syvyysarvoina kohteiden reunoilla. Koska järjestelmä on luotu lisätyn todellisuuden sovellusten tueksi, se ei tuota mittaussovellusten näkökulmasta optimaalista aineistoa.
Havaituista puutteista huolimatta laitteella on potentiaalia myös mittauksiin, koska tuotettuun aineistoon tallentuu suuri määrä tietoa. Saman tiedon kerääminen tavanomaisin menetelmin voisi olla haastavaa, aikaa vievää tai kallista. Sopivilla prosessointi- ja suodatusmenetelmillä havaittuja ongelmia voidaan myös karsia. On myös mahdollista, että tulevaisuudessa laitteen käytettävyys mittauksiin paranee laitevalmistajan rajapintojen kehittyessä.

Avainsanat Apple iPad Pro, LiDAR, syvyyskamera, tarkkuus, hajonta

Contents

1 Introduction 8
2 Background 10
2.1 Light detection and ranging 12
2.1.1 Time-of-flight 13
2.1.2 Flash LiDAR 14
2.2 Positioning 15
2.2.1 Inertial Measurement Unit 16
2.2.2 Visual-inertial odometry 16
2.3 Error analysis 17
2.3.1 Depth map quality assessment 18
2.3.2 Point cloud quality assessment 19
3 Data and methods 20
3.1 Devices used in data collection 20
3.2 Test set-ups 21
3.2.1 Illumination unit test 21
3.2.2 Testing importance of RGB camera in depth map creation 22
3.2.3 Depth measurement repeatability 23
3.2.4 Depth measurement consistency over a depth map 25
3.2.5 Depth measurement repeatability for different materials 26
3.3 Software and analysis 27
4 Results and discussion 28
4.1 Illumination unit pattern and intensity distribution 28
4.2 Importance of RGB image in depth map creation 31
4.3 Range measurement repeatability 33
4.4 Depth measurement consistency over a depth map 38
4.5 Depth measurement repeatability from different materials 41
4.6 Summary of findings 44
5 Conclusions 46
A. Data processing scripts 52

Preface

Special thanks to Arttu Soininen for the topic, devices, and deep insights to the practice of the LiDAR technology throughout this work. Thanks to Doctoral Candidate Heikki Kauhanen and Assistant Professor Matti Vaaja for their active advice. In addition, I want to thank Isabel for her support during this work.

Espoossa 22.11.2021

Antti Järvenpää

Symbols and abbreviations

Symbols

d	[m]	Diameter of a beam
f	[pix]	Focal length of a camera
h		Size of a dimension
P	[W]	Radiant power
R	[m]	Range
s		Conversion scale
t	[s]	Time
θ	[rad]	Diverging angle of a beam
v	[m/s]	Speed of light in a medium
Φ	[W/m²]	Density of radiant power

Lower indices

im	In image space
t	Transmitted
tar	On target

Abbreviations

API	Application Programming Interface
AR	Augmented Reality
FOV	Field of View
IMU	Inertial Measurement Unit
IQR	Interquartile Range
IR	Infrared
LiDAR	Light Detection and Ranging
RANSAC	Random Sample Consensus
RGB	Full colour – red, green, and blue
SLAM	Simultaneous Localization and Mapping
TLS	Terrestrial Laser Scanner
TOF	Time-of-flight
VIO	Visual-Inertial Odometry
VO	Visual Odometry

1 Introduction

As data collection technology becomes more mature, it will most likely develop cheaper, more accurate, and/or spread to new applications. Recent interest in three-dimensional sensing and development of laser optics have gained help from each other, and technologies are spreading more widely to applications on various platforms (Dummer et al., 2021).

Overall, three-dimensional sensing is used to support various applications, like autonomous driving, extended reality, and conventional measurements and modelling (Dummer et al., 2021). In measurement applications, the product might vary from topographical model of the Earth's surface (Shan & Toth, 2018, 2) to documentation of cultural heritage as a digital twin of a statue (Murtiyoso et al., 2021). Naturally, the requirements for a data collection system depend on the application and desired product. However, many applications probably would benefit from a solution capable to collect, validate and even process the data *in situ*, enabling improvements on the fly without loss of valuable time.

In 2020, Apple released a new generation of its popular iPad tablet devices. The top model of these devices, called iPad Pro, featured a flash LiDAR sensor in the back of the device alongside of the camera sensors (Apple Inc., 2020a). The sensor was built in with augmented reality (AR) applications in mind but having technology available naturally raises interest to use it for other applications as well. As a platform for data capture, a tablet computer could be rather beneficial as it comprises elements for processing and visualization while retaining mobility.

Metrology as a term originates from Greek language, and, by definition, means scientific study of measurement. In this work the Apple iPad Pro (2020) depth sensing system is characterized from metrological point of view. The characterization includes investigation of error sources and their magnitudes, and investigation of features affecting the use of the system in measurement applications.

In general, ability to physically measure distances can improve data collection capabilities by providing more consistent and accurate data flow compared to shape acquisition from images only (Ingman et al., 2020). This raises a need for independent evaluation of the accuracy and precision of the system. Earlier, the use of iPad as data collection device has been evaluated by a few groups: Vogt et al., Heinrichs and Yang, Gollob et al., Murtiyoso et al., and Luetzenburg et al., all in 2021.

These previous studies have examined the sensor usability in practical applications, rather than taking more general, metrological approach. Each group has selected their own use case and performed so called end-to-end testing for the system. Examples of the use cases are applicability in reverse engineering (Vogt et al., 2021), documentation of cultural heritage (Murtiyoso et al., 2021), and geoscientific measurements of a coastal cliff (Luetzenburg et al., 2021).

All these earlier studies well point out common issues with the technology. The most significant issues these groups pointed out were lack of capabilities to preserve details (Vogt et al, 2021), point-to-point noise (Murtiyoso et al., 2021), and drift in positioning (Gollob et al. 2021; Heinrichs & Yang, 2021; Luetzenburg et al., 2021). Nevertheless, the sensor itself remains unevaluated. In addition to complete system testing, depth sensing capabilities of the sensor should be evaluated separately to get a better understanding of the current state of the sensor technology. This is important as the choice of processing algorithms most likely has a strong impact on the product, and thus even big leaps in the technology might be

possible in future years. Some of the tests were also performed in a sample environment not suitable for the system, conclusions about the accuracy in such challenging set-up may not be relevant.

Also, discussion about the weak points leading to inaccurate results is still missing. To help users get most out of the device and technology, it is good to recognise the problems and avoid them. This way the result may be improved to acceptable level without a need to invest in more expensive solutions to get the same result with that small improvement in reliability or accuracy.

The aim of this work is to examine the depth sensing accuracy of the prementioned system focusing on the depth sensor and examine characteristic features of the system that affect the data collection. Due to the focus on depth sensing, device is held still during data captures, and location tracking accuracies of the system are not investigated, though technologies and sensors included in the device will be described in the literature review part of the work. Location tracking technologies are included in theoretical discussion as those may have significant effect in the accuracy of practical applications. Thus, it is important to understand uncertainties of positioning and remember that those together with depth sensing solution and other possible error sources together propagate the total error (Berendsen, 2011, 37–38; Vermeer, 2019, 19–20).

The experiments in this work focus on the precision of the sensor while the absolute accuracy of the system is discussed less. The experiments investigate the depth measurement spread as function of distance and coherency of data within a depth map. Results help to decide practical applications for the system and even more to understand what its limitations are.

The next section of this work reviews the background of the topic: characteristics of the technology included in the device and error analysis. The third section begins the experimental part of this work, and introduces the devices and methods of data collection and processing used in the experiments. The fourth section presents the results together with discussion, and the last section draws conclusions about the work as a whole and gives some thoughts for future work and development of the field.

2 Background

The aim of this section is to provide understanding of the examined technology. The beginning of the section focuses on the technology on a large scale, explaining how it has developed during the past years, what are the applications and what are alternatives for it. After these general thoughts, the following two subsections explain in more detail the technical solutions of the sensor focusing on their contribution to the measurement error. Finally, subsection 2.3 focuses on the error analysis, explaining how it can be performed on the data products collected with the system.

The technology examined in this work is one method for obtaining three-dimensional information from natural environment. Several different technical approaches exist for obtaining the same information (Giancola et al., 2018, 2), and the taxonomy of these methods is presented in figure 1 below. Taxonomy of the subclass of optical methods is presented in the following subchapter in more detail.

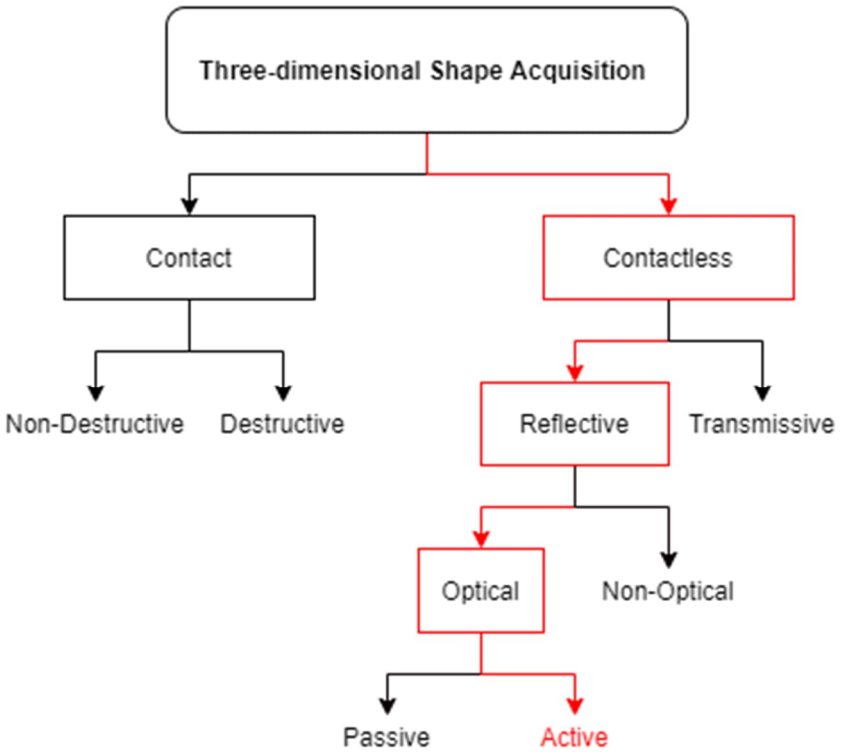


Figure 1. Taxonomy of shape acquisition methods with the examined technology highlighted in red (Adapted from Giancola et al., 2018).

As figure 1 shows, the examined technology does not require physical contact to the target objects. Instead, the technology is based on sending light pulses to the target scene and observing the reflection. First implementations applying this approach have been proposed in the late 80's (Beheim & Fritsch, 1986). Since then, the technology has gone through huge development, e.g., the range of measurements has increased from few meters (Beheim & Fritsch, 1986) to even kilometres (Shan & Toth, 2018, 60).

One of the development steps since the beginning of this millennium has been to apply array sensor in measurement to gather image like depth map at a time instead of individual points Lange & Seitz, 2001). Depth map collection improves possibilities of the technology in many applications. Compared to capture of individual points, depth map capture increases the data collection rate, can capture snapshots of moving objects, and adds weak topology to the observations (Weinmann, 2016, 23; Remondino & Stoppa 2013, 6–7). However, the need to illuminate the target scene when collecting a depth map significantly increases the need of illumination power. This leads to either higher power consumption or lower depth range (Lange & Seitz, 2001). Also, though the captured depth map is like common digital image, the resolution is commonly far behind conventional colour (RGB) images in practice (Remondino & Stoppa, 2013, 2). However, the intensive research and development of applications, such as autonomous vehicles, throttles the development of the sensor technology and price, size and availability of the sensors improves all the time (Chen et al., 2021).

Light detection and ranging used to be an expensive and bulky technology (Lange & Seitz, 2001), but nowadays it has been adopted to everyday use in wide range of applications (Shan & Toth, 2018, 60–62, 86). Dependent on the sensor type, it can be applied to measurements and sensing of environment in various cases. Figure 2 below presents some common applications of the technology.

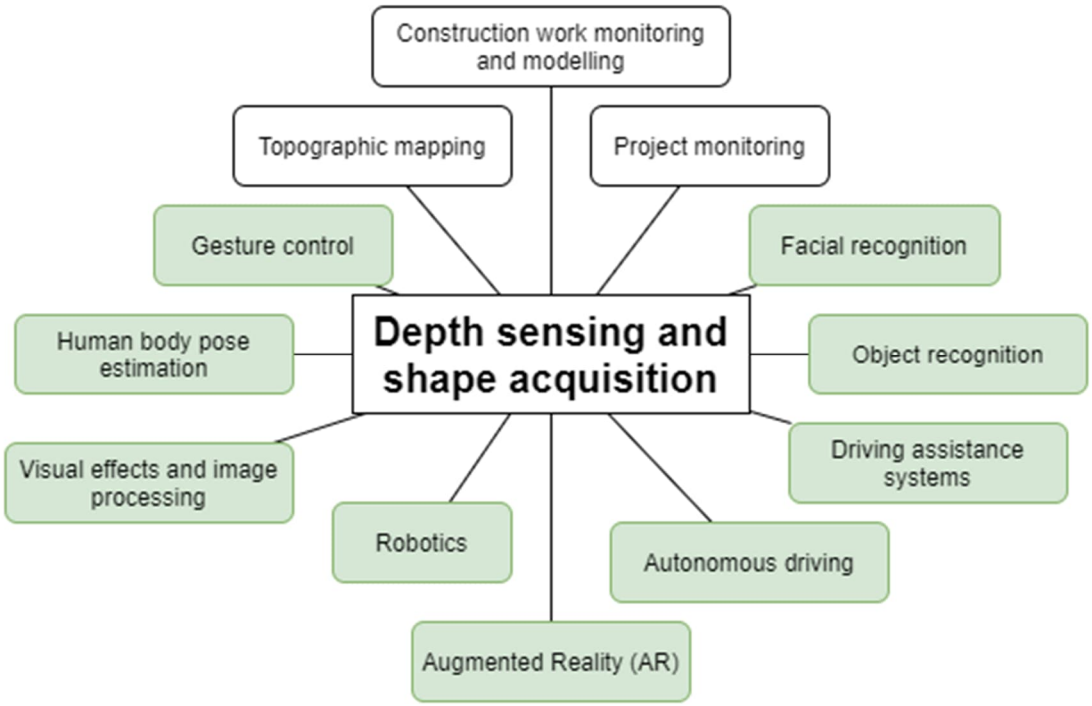


Figure 2. Example applications for three-dimensional measurements. Applications that may advance from depth maps over single measurements are highlighted in green (Adapted from Chen et al., 2021; Dummer et al., 2021; Grzegorzek et al., 2013, 54; Shan & Toth, 2018, 60–62).

In the figure, applications where flash LiDAR –type of sensor can provide advantage over scanning sensor are highlighted. In general, shape acquisition methods can be used to gather

information for later use, or to enhance interaction to environment in real-time applications. From these two alternatives, the flash LiDAR technology is beneficial especially in real-time applications, as it records continuously depth map from the scene instead of single point information (Weinmann, 2016, 23). These applications also withstand better the drawbacks of the technology, such as lower precision, higher noise ratio or low range (Grzegorzek et al., 2013, 54).

In general, the first widespread application of laser ranging has been surveying, starting in the 1960s, and scanning LiDAR devices were adopted from laboratory to practical use in the middle of 1990s (Shan & Toth, 2018, 2). Since these days, the technology has developed allowing it to spread to various applications, such as Microsoft Kinect V2 game controller, which might be the most famous time-of-flight range camera (Giancola et al., 2018, 41).

2.1 Light detection and ranging

Light detection and ranging (LiDAR) is an active method for performing range measurements, where the range estimation is based on physical properties of observed light (Vosselman & Maas, 2010, 1–2). Being an active method means that the system includes an illumination source, and is not reliable on external light (Weinmann, 2016, 22–23). Figure 3 below illustrates the basic taxonomy and some examples of sensors and their alternatives. These alternative technologies are not covered in this work.

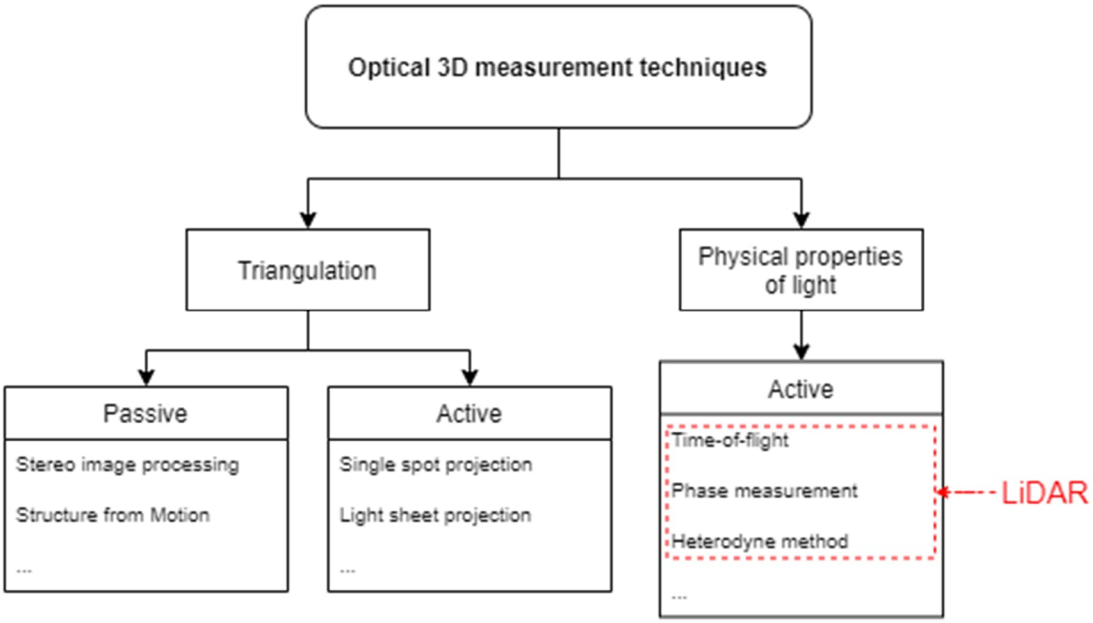


Figure 3. Taxonomy of three-dimensional measurement techniques and some examples of alternative techniques according to Vosselman and Maas (2010) and Weinmann (2016).

In all sensors implementing the LiDAR approach the idea is to send laser pulse to the environment, observe the scattered beam and derive the distance from its properties. The distance together with the orientation information of the sensor leads to three-dimensional coordinate information of the observed point.

To derive the distance to the target, there are three alternative ranging techniques available: Time-of-flight (TOF) measurement, phase measurement and heterodyne method (Shan & Toth, 2018, 14). Two first of these methods are most used approaches (Vosselman & Maas, 2010, 2), whereas the latter one is rather in prototype phase than in wide use (Shan & Toth, 2018, 21). Instead of using physical properties of light in ranging, optical measurement could be based on triangulation as well (Vosselman & Maas, 2010, 2). Triangulation based systems do not use travel time of light to estimate range but observe distinct points from the scene and compute their location with help of trigonometry (Giancola et al., 2018).

The methods listed have rather different approaches and properties. For example, phase measurement can reach better resolution and data rate compared to TOF, but is limited in range (Vosselman & Maas, 2010, 8). On the other hand, the heterodyne method tolerates noise and penetrates better, which comes with the cost of complexity (Shan & Toth, 2018, 21). The following section will focus on the TOF method, which the iPad sensor makes use of. The other methods are not covered in this work.

2.1.1 Time-of-flight

Respective to its name, TOF method is based on measuring the transit time of a light beam in a medium and deducing the distance to the object as the velocity of light is known (Shan & Toth 2018, 14; Vosselman & Maas, 2010, 3). Here, figure 4 illustrates the key elements and the basic idea of the method.

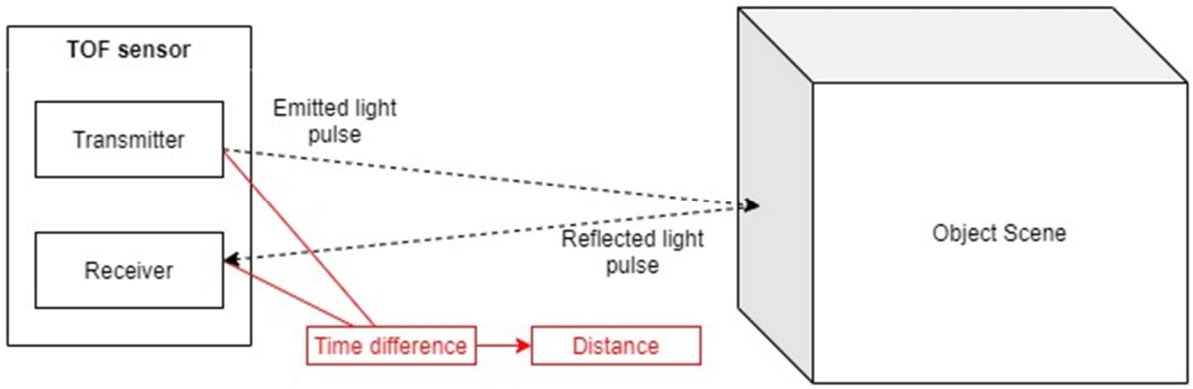


Figure 4. Basic principle and key elements of a TOF system.

In the left part of the figure, the schematic of a TOF sensor shows two key components of a such ranging device: transmitting illumination source, that could be a laser emitter, and receiving optical sensor which detects the reflected laser pulse. As the transmitter sends a beam to the scene, time measurement starts. Respectively, it ends to the moment of return of the laser beam, i.e., the moment the optical receiver detects the reflected pulse. As the velocity of light is a known constant, the distance to the object causing reflection can be computed from the travel time of the laser pulse using following formula (Shan & Toth 2018, 3):

$$R = \frac{\upsilon\, t}{2}, \qquad 1$$

where R denotes the range, υ refers to the speed of light in the medium and *t* to the observed travel time of the light pulse. This equation expects the path of light pulse to be same back and forth from the light source to the sensor. Following from this, some adjustments might be required in practice depending on the construction of the apparatus, e.g., if the light beam is redirected with mirrors or optical elements within the system or if the light source and the receiver are placed next to each other and the light path is slant between them as it most likely happens with a flash lidar sensor.

Ranging with TOF method requires some choices of technical approach. Different choices might lead to different result and may thus be interesting from the data quality perspective.

In this context, we discuss the challenges related to the sensing of the incoming pulse. The detecting sensor records the intensity of incoming light. Instead of being originated from the related illumination source, the incoming photons might be background noise originated for example from the sun (Chen et al., 2021). In addition, one pulse might result in multiple returns (Shan & Toth, 2018, 206–207). Both these highlight the importance of pulse detection solution, which might, e.g., be based on the extraction of highest peak or leading edge of the incoming pulse (Shan & Toth, 2018, 247–251). However, due to the data processing solution that is built in the examined system, it is not possible to draw reliable conclusions about the pulse detection solution of the system. The processing solution distorts the data too much, covering important details. Hence, the differences between pulse detection solutions are not examined in this work..

2.1.2 Flash LiDAR

The technique discussed so far can capture one point at a time from the environment. To get a complete cloud of points, new laser beams should be directed to the object environment all the time, while also moving the device or changing its orientation. For conventional terrestrial laser scanning (TLS) systems this is normally performed by rotating the system or turning the beam direction with the help of mirrors or prisms (Shan & Toth, 2018, 32). However, physically moving parts may be difficult to fit in a small device such as a tablet, and only moving the device manually may be slow from data collection point of view. This problem may be overcome by illuminating the target completely instead of one small dot, and then observing the reflected beams with an array sensor. This type of sensor is commonly called flash LiDAR (Vosselman & Maas, 2010, 19).

Whereas a conventional laser scanner sensor produces single range measurement at a time, flash LiDAR sensor records a depth map of the object scene. A depth map is comparable to an image, but each pixel stores the scene's depth value covered by respective image area. An example of a depth map is represented in figure 5.

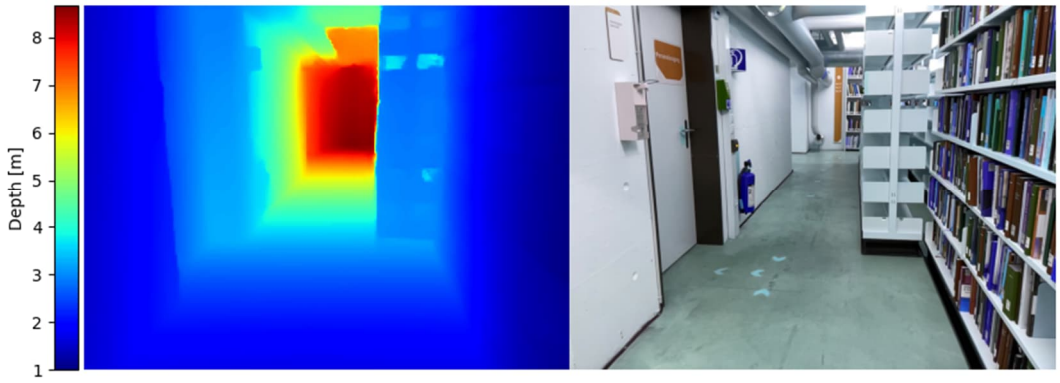


Figure 5. An example of a depth map and an RGB image of the corresponding scene.

A notable limitation of a flash LiDAR sensor compared to a similar scanning LiDAR system is its lower maximum range: a flash LiDAR system must illuminate a wide scene whereas a scanning system can reach better power density by focusing the power to a single dot. The power density at the target is defined by equation

$$\Phi_{\text{tar}} = \frac{P_{\text{t}}}{\frac{\pi}{2}(\theta_{\text{t}} R + d_{\text{t}})^2}, \qquad 2$$

where Φ_{tar} denotes the power density at the target, P_{t} the power of transmitted light, θ_{t} is the diverging angle of the light beam, R is the distance between sensor and the object and d is the diameter of transmitted beam (Shan & Toth, 2018, 23). Here, as the diverging angle is raised to the power of two, it has significant effect to the actual power that can be observed. To tackle this problem, Apple has developed a flash LiDAR sensor that does not illuminate the whole scene, but rather a set of points to the scene.

As discussed already in the introduction, the LiDAR depth sensor has been added to the devices mainly to improve the performance of the device in AR applications. To keep AR experience smooth and realistic, the data flow to the applications must be consistent and continuous to not lose orientation or environment tracking. With depth map data, this could mean e.g., that the system interpolates depth values for some pixels, if range information cannot be obtained for the whole image at some time. This could happen e.g., if the scene depth is outside the effective range of the sensor.

With Apple devices, corresponding corrections and interpolation are performed for all depth frames, according to their materials (Apple Inc., 2020b). The materials declare that the actual sparse depth data is processed together with the RGB-image from the scene to produce a corresponding product in better resolution. This arrangement is also used to improve power efficiency of the device: instead of illuminating the whole scene, only sparse set of beams is enough to provide some data to the detector (Oggier, 2020). In the experimental part of this work, feasibility of this solution is tested. In experiments, depth maps are captured both with and without the assistance of RGB camera, and results are compared.

2.2 Positioning

A flash LiDAR sensor captures distance and direction from the sensor to a set of points at a time interval. To combine observations at different timesteps to a single set of points, the

relative orientation of the sensor between these timesteps must be known (Vosselman & Maas, 2010, 22–23). Further on, if there is interest to reference this set of observations geographically or to some existing model, the orientation between the points and the reference must be recorded. This subsection focuses on describing sensors and techniques used in the examined tablet device. The text will not focus much on geographical localization, as it is not very usable in the derivation of a dataset.

To register individual depth map frames, i.e., solve relative orientations between them and combine the information to a single dataset, alternative approaches are available. One available method is to keep track on the device orientation and combine frames in real-time (Shan & Toth, 2018, 180). Another alternative could be to collect batches of depth images from various stationary perspectives and do the registration in post-processing (Weinmann, 2016, 77). Both methods benefit, if the real time positioning accuracy is good, as it leads to better initial values also for the latter approach (Shan & Toth, 2018, 180; Weinmann, 2016, 77).

Accurate real-time localization in an arbitrary environment is a challenging task, that currently cannot be solved reliably with a single sensor (Shan & Toth, 2018, 180): satellite positioning has blind spots below obstacles and is not precise enough, inertial measurement units (IMU) suffer from drift and optical motion tracking, i.e., motion tracking from image flow, may lose the tracking if the environment does not have sufficient texture for feature extraction. However, the problem can be overcome by combining multiple methods. In Apple devices, the real time localization is performed with a method called visual-inertial odometry (VIO), and the information is shared for apps through an application programming interface (API) (Apple Inc, 2021).

2.2.1 Inertial Measurement Unit

Inertial Measurement Unit (IMU) is a movement tracking system, which usually comprises accelerometer and gyroscope (Poddar et al., 2017). Accelerometer is a sensor for measuring velocity changes, whereas gyroscope provides information of rotations. The measurements of the IMU may be supported by magnetometer and temperature sensor (Poddar et al. 2017).

Inertial sensors apply a few technical approaches that result in different accuracy grades (Han et al., 2020). Most common type of sensors in consumer electronics are microelectromechanical system (MEMS) based sensors due to their manufacturing capability and high accuracy-price ratio (Poddar et al., 2017). Typical range of bias for consumer grade gyroscopes is one degree per minute, and for accelerometers 1 mm/s^2 (Han et al. 2020). If biased observations are used in positioning during data collection, the resulting trajectory and corresponding point cloud may become distorted with bending and stretching parts. This problem is commonly referred to as drift in literature. Drifting of IMU observations means that inertial positioning is alone not a robust enough positioning method for continuous data collection (Shan & Toth, 2018, 180–181).

2.2.2 Visual-inertial odometry

VIO is not a single algorithm, but it has several implementations (Schubert et al., 2018). The core idea of VIO is to combine motion estimation by inertial measurements and feature tracking from images (Leutenegger et al., 2015; Li & Mourikis, 2013). The approach has

some major benefits. Firstly, using image feature keypoints helps to filter out low frequency drift related to inertial measurements (Leutenegger et al., 2015). Secondly, inertial measurements help image feature tracking to deal better with untextured environments and rapid motions (Schubert et al., 2018).

In general, the approach of visual odometry (VO), is to track motion by updating the state of pose and location from changes in keypoint features between two consecutive images (Nistér et al., 2004). So, compared to simultaneous location and mapping (SLAM), the key difference is that the feature observations are not stored, but image features are tracked only in short term. In VIO, the fusion of inertial measurements to optical motion tracking has several alternative approaches, and the choice is about balancing between desired accuracy and computational cost (Leutenegger et al., 2014). For example, a loosely coupled system combines separate motion estimates from these two sources, whereas a tightly coupled system jointly optimizes the result, having higher complexity but lower loss of information (Leutenegger et al., 2014; Li & Mourikis, 2012).

As VIO is a complex technology also dependent on the properties of the environment, it is rather impossible to state a globally valid estimate of algorithm accuracies, but comparison and evaluation can be performed with benchmark data (Schubert et al., 2018). For modern implementations, 0.04 ° standard deviation for pose (Li & Mourikis, 2012) and 1% relative error for position (Leutenegger et al., 2014) can be expected, though the results may vary dependent on the implementation and environment (Schubert et al., 2018).

2.3 Error analysis

The beginning of this section includes discussion about the term “error” in general. The text will sharpen what is meant by error and what types of error there are. The latter two subsections discuss approaches to measure these errors. The discussion will focus on the evaluation of data captured with the examined device, and may exclude unsuitable methods.

Error may occur in various forms. It may appear as random deviation, systematically e.g., by uncalibrated measurement devices, or as erroneous outlier recorded by mistake (Berendsen, 2011, 18–19). In addition, the behaviour of the error of two variables may be correlated, which means that their magnitude follows each other (Vermeer, 2019, 30). These types of error have been illustrated in figure 6.

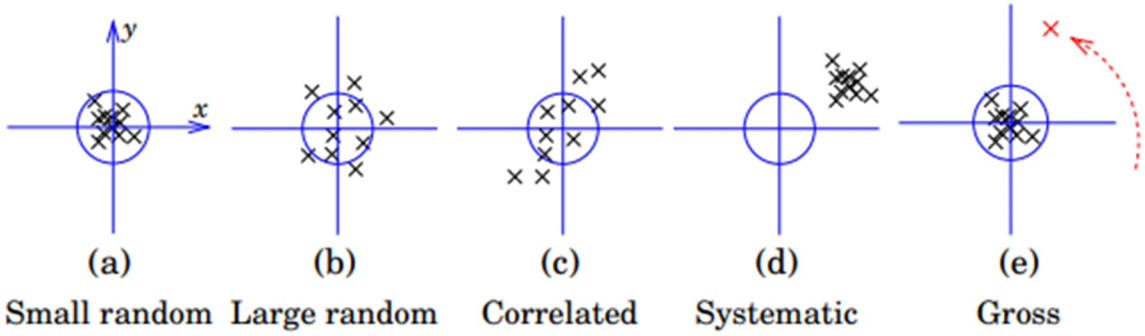


Figure 6. Illustration of different error types (Vermeer, 2019, 30).

Above, the variants a and b illustrate one form of random deviation called Gaussian noise. In example c, the error the error of two variables is correlated with each other, as its magnitude follows a trend. In case d, the error is systematic. Error being systematic indicates that

its direction is constant. This leads us to the difference of accuracy and precision, which will be discussed in the following paragraph. The last error variant displays gross error, an outlier point, that might be caused by e.g., malfunction of measurement devices or people walking in the target scene while laser scanning measurement was ongoing. Nevertheless, this might even be completely true value, even if completely different from the trend. Such cases are commonly referred to as *black swan* in literature.

While discussing different appearances of error, it is also important to consider the difference of *accuracy* and *precision* (Vermeer, 2019, 28). Even if measurements are very well repeatable, i.e., those deviate extraordinarily little from each other, they can differ from the ground truth a lot. In such situation, the measurements are precise, but not accurate. Case d in figure 6 could be considered as an example of such situation. On the other hand, some measurement method might result in different values over time, but when the observed numbers are averaged, the mean value corresponds well the reality. Now, the method is accurate, but not precise. Case b in figure 6 represents such a situation: the observations are more sparsely distributed than in case a or d, but the centre of mass of the observations is closer to the actual value, which is not the case in variant d.

To address these different forms of unreliability, various alternative approaches exist. One way is to inspect the error analytically, dividing the whole data collection process to small parts and combining the uncertainties of these to the total error by applying the law of error propagation (Berendsen, 2011, 135–137). Another alternative would be making empirical measurements and analysing accuracy and reliability of the system either by statistical methods (Berendsen, 2011, 151–153) or by comparing to reference information (Ingman et al., 2020). To improve comparability of uncertainties obtained with different methods, frameworks and standards guiding the process, such as GUM, exist (JCGM, 2008).

In the analytical approach, the former one mentioned in the previous paragraph, all uncertainties related to the data acquisition process need to be mapped together with the methods applied. All these are required to produce a convincing result not neglecting any sources of error. For a complex system of many integrated sensors, the complexity of the analysis grows rapidly. The examined application integrates multiple sensors (e.g., camera, flash LiDAR and IMU) and applies complex artificial algorithms (such as VIO), from which even part is proprietary, so the analytical error is not computed in this work. However, the earlier subchapters 2.1 and 2.2 have tried to assess the possible error magnitudes of the system parts to understand their significance. Following two subsections highlight some available methods and approaches for evaluating the error from empirical measurements, which will be the approach in the experimental part of this work.

2.3.1 Depth map quality assessment

As discussed earlier in the subsection 2.1.2, depth map is an image-like data structure, which stores scene depth values in each of its pixels instead of colour. Thus, quality assessment methods like ones used with conventional optical images could be applied for depth maps as well. These methods may be subjective, based on visual interpretation, or objective, providing computational quality index of the image under assessment, such as structural similarity index (Wang et al., 2004). In this work, visual interpretation and comparison is used.

However, quite different approach could be available also, as reference information for the analysis could be produced in other ways than optically, e.g., by making physical measurements of the target and modelling the scene (Ingman et al., 2020; Murtiyoso et al., 2021). Further on, analysis could be performed statistically from a series of identical measurements (Giancola et al., 2018). In this work, the statistical approach is used more, and comparison to reference data is used only on some occasions to validate depth data.

2.3.2 Point cloud quality assessment

The error in point localization during the data collection might be related to acquisition system, environment, target scene or object related issues (Weinmann, 2016, 26). One way for estimating the reliability of the observations is to use physical properties, and filter low return intensity points or points on sharp edges from the data (Weinmann, 2016, 27–28). Another possibility is to evaluate the quality visually, leading to subjective rating of data quality (Alexiou & Ebrahimi, 2018). third, the data can be evaluated against reference computing, e.g., point distances to reference cloud or mesh or computing geometrical similarity to reference (Alexiou & Ebrahimi, 2018; Ingman et al., 2020; Murtiyoso et al. 2021).

In the experimental part of this work, analysis is mostly performed on data in depth map format. However, some experiments include evaluation of data quality in point cloud format through a visual analysis approach. In the experiments, point cloud quality assessment is barely applied, as the focus of the work is on the depth sensor, and other parts of the system are not the main interest.

3 Data and methods

As discussed already in the introduction, the idea of this work is to test the technology in such environments, where it might be used in practice. This means not focusing on small features, that the system is not able to preserve, but on capturing data from bigger objects. The environment might have complicated shapes though, as in such environments the measuring capabilities of point cloud might be preferrable over physical tape measurements.

The following three subsections describe in detail the devices used in this benchmark, environments in the set-up, and the software and methods used in processing and analysis of the data, respectively.

3.1 Devices used in data collection

In this work, sample datasets are collected with Apple iPad Pro (2020 model), which is the device under analysis. Apple iPad Pro is a tablet device, running on Apple iPadOS, and comprising a TOF range sensor (Apple Inc., 2020a). A picture of the system's appearance is presented in figure 7. The device used in the test is not the only available mobile device comprising flash LiDAR technology, but also other manufacturers have adopted the technology to their devices. Some examples of devices comprising the technology are Samsung Galaxy S20+, Samsung Galaxy S20 Ultra (Samsung Electronics Co. Ltd., 2020), and Lenovo PHAB2 Pro (Lenovo Ltd., 2016). These other devices are not used in this work.

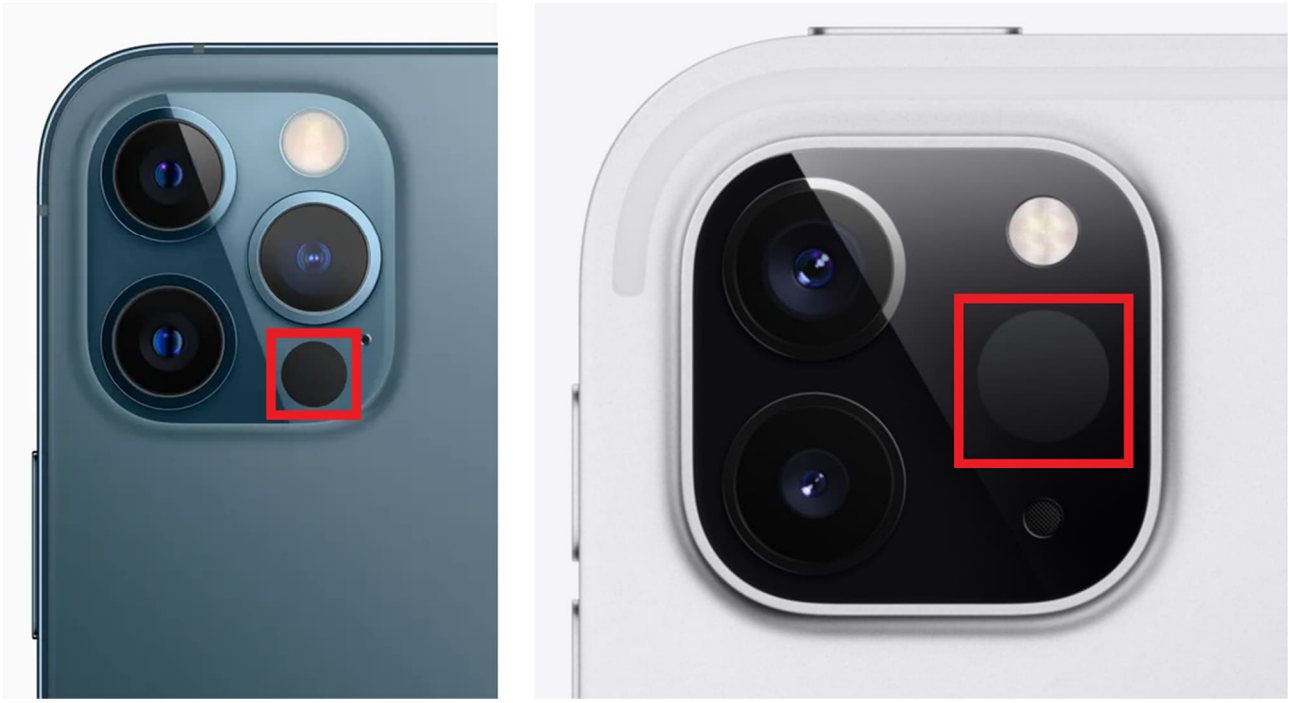


Figure 7. LiDAR sensor location on Apple iPhone Pro 12 (left) and iPad Pro 2020 devices. Only the latter is used in this work.

In most of the tests, the data is analysed using statistical methods without any actual reference data from other sources. In some experiments, e.g., when examining the relationship between observation distance and measurement deviation, rough reference values are

obtained with tape measure. These values are not used for precise evaluation of the absolute accuracy of the system as these values cannot be considered accurate enough for such analysis.

Although Apple uses term LiDAR to promote the devices having the sensor, the devices do not natively have any software to collect data and neither do they provide access to the raw depth information (Vogt et al., 2021). Instead, only processed data is available, and a third-party software must be installed and used to save it. The data processing workflow fuses RGB images and flash LiDAR observations together (Apple Inc, 2020b). One Apple patent (Norman et al., 2019) describes a method for the fusion with a "trained machine learning model such as a neural network", and it is very likely that this approach is used in the system. Some alternative third-party data collection software found in the Apple App Store were SiteScape, Heges and Stray Scanner.

In this work, Stray Scanner is used for capturing depth data. This app was preferred over other alternatives, as it provides access to as low-level data products as possible, as it does not process data from iPad APIs, but stores the data as it is. This is preferrable during the research as it reduces the effect of used application in the result. The data that the application stores comprise depth maps, corresponding confidence maps, RGB camera feed, intrinsic parameters of the camera and the trajectory data storing the orientations of the system during the data collection session.

3.2 Test set-ups

This subsection describes different experimental set-ups used in this work. In the beginning, some choices common to all experimental cases are pointed out and explained. Then, following five subsections describe characteristics of individual experiments.

In all experiments, if not otherwise stated, RGB camera sensors of the tablet device are covered with opaque cardboard. The cover is attached with two-sided adhesive tape to prevent the system from using colour images to refine depth map information.

Another practical choice in this work relates to data processing: from all depth map series, a short, one second period of data is removed to prevent artifacts from user interaction to end up in the data. Such artifact could be caused by the user, when data collection is started, and device shakes unintentionally from touch.

3.2.1 Illumination unit test

The aim of this experiment is to define the distribution of the illumination power of the laser emitter. The illumination power has important role defining the quality of the depth map produced by the sensor, as higher reflection power helps the detection of actual pulse from background noise. Thus, evenly distributed illumination power may improve the overall quality of a depth map and make depth values consistent.

In addition, the opening angle of the laser pattern is defined during the test and compared to the field of view of the sensor. The opening angle is defined by placing the system pointing towards a wall from a known distance, defining edges of the laser pattern on the wall with IR camera and measuring the dimensions along central axis both horizontal and vertical direction. Then, angles are calculated with simple trigonometry

$$\alpha = 2\tan^{-1}\frac{h_{\text{tar}}}{2R}, \quad 3$$

where α denotes the angle, h_{tar} the width of the light pattern on the target wall, and R the distance from the system to the target wall. The width is either horizontal or vertical width of the pattern, chosen respective to the direction of the desired opening angle. This equation expects the system to point exactly towards the wall, but the error from possible skew is considered small.

The corresponding field of view is defined from known intrinsic parameters of the sensor. As intrinsic parameters the ones reported by the iPad API through data capture software are used. The calculation is performed using equation

$$\text{FOV} = 2\tan^{-1}\frac{h_{\text{im}}}{2sf}, \quad 4$$

where FOV denotes the field of view, h_{im} the width of the image in pixels, s scale value for the focal length, and f the focal length in pixels. Scale for focal length is required as the system reports the focal length in resolution of RGB feed, and the resolution of depth maps is different. Again, the width must be either vertical or horizontal depending on which field of view angle is computed.

In this experiment, the device was set on tripod facing directly towards a wall in a dark room illuminating the wall and IR camera was attached above it. Then, the camera was set to take ten images of the illumination pattern. After this, the tablet was rotated so that the edge of the illumination pattern pointed directly towards the wall, and the camera pointed to the same direction as in first case. The image capture was repeated with same settings (same shutter speed, ISO value and aperture) as in first case to make the image sets comparable. Shutter speed was set relatively slow, two seconds, to prevent the data capture only from empty time between laser pulses.

In processing phase both ten image sets are averaged, smoothed with gaussian method and individual laser spots are segmented from the image. Maximum intensity values from smoothed pictures are used to represent the illumination intensity of each spot. For interpreting the illumination power distribution in single scene, the detected spots are plotted. To help comparison between cases where device is facing directly towards the wall and the other where the device is rotated, intensity values are interpolated to a continuous field and plotted with common colour scale normalization.

3.2.2 Testing importance of RGB camera in depth map creation

The aim of this experiment is to deduce if the system can produce valid information when the camera is covered. The main idea is to help the decision of whether to let the system work in further experiments as it is designed, or to cover the camera to help the work focus on evaluating the LiDAR system alone.

The experiment is performed as follows: the system is directed towards a built test scene in a dark room. Lights are turned off to make it impossible for the system to help depth map creation with image information. The data collection is turned on and continued as such for ten seconds. Then, lights are turned on while continuing the data collection. The data is collected still for other ten seconds . The data collected during the transition between light conditions is excluded to prevent immature data products from affecting the result.

In this experiment the built test scene contains three different test targets: a simple geometry, a detailed object, and a picture of some geometry. As a simple geometry, a block of notes is used. The aim is to provide a target that for sure is hit by the laser light, and produces a response to the range image. As second target, a couple of pencils are placed into a cup. Here, the idea is to make the target dispersed so that the system cannot get laser hits on every detail of the target. As the third target, a picture of a geometrical feature is attached on a wall. The intention is to see if the system can be tricked or if it relies on the LiDAR information. An illustration of the test scene is present in figure 8.

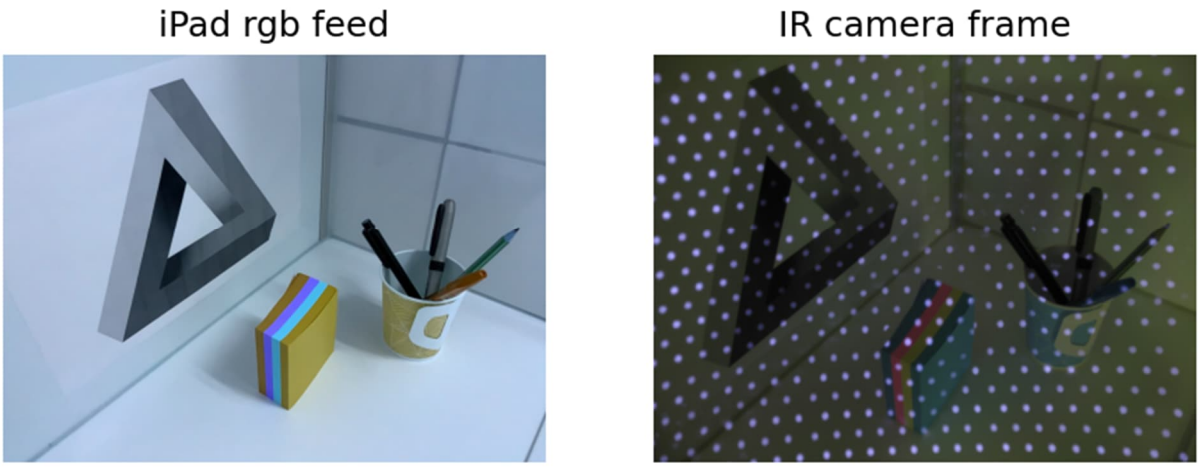


Figure 8. A picture of the built test scene and illustration of the laser pattern in it.

Figure 8 shows the arrangement of objects in the scene during the test. The left frame of it also shows the distribution of laser dots during the measurement.

After the data collection, data series corresponding to the two cases are extracted and visualized Analysis of the information is performed visually by comparing the visualizations.

3.2.3 Depth measurement repeatability

The aim of this experiment is to find out how precise the range measurement generally is, i.e., how big variations exist between repeated measurements. The measurement is repeated on multiple distances, and thus, the effective range of the device is determined simultaneously. In measurement applications, the importance of the precision depends on the overall capture method: if the device is placed to collect a series of observations from the scene, redundant observations can be used to remove random noise (Vermeer, 2019, 30). Nevertheless, if all observations are used as individual points, the noise remains in the data and affects the result.

In this experiment, series of depth maps are collected from the object to define the spread of observations statistically. The data collection is repeated on multiple distances from the object to define how the distance affects the precision. The expectation is that the longer the range is, the larger the spread will be. On longer range the illumination power spreads to a larger area, decreasing the signal-to-noise -ratio most likely leading to less reliable results.

The data collection is performed in two separate parts. The first part focuses on the overall view of the data collection, whereas the other investigates the accuracy in short range. Short range measurements also aim to reveal the lower limit for the effective range of the system.

These two measurements are collected separately to ensure better accuracy of the reference measurements.

The first measurement is performed in a parking garage. Series of depth maps are collected from different ranges and descriptive statistics of spread in observations are computed. An overview of the measurement set-up is presented in figure 9.

Figure 9. Overview of the experimental set-up in the parking garage.

The figure above shows example of a measurement station, where one series of depth maps is collected. Station locations are defined with the help of tape measure, and stations are placed with half a meter interval. The device is attached to a tripod and turned pointing directly towards a wall on each station. A sheet of paper is attached to the wall and used as a target. The idea of using paper is to improve repeatability of this test in different environments, as reflective properties of the paper remain the same.

In this environment, the level of background radiation is rather low as there are no windows and only a few fluorescent tubes are lit during the data collection. On each measurement station, it is ensured with help of IR camera, that the laser light illuminates at least one dot on the target paper. At each station, the data is collected for ten seconds, which leads to 480 pcs. depth map sample size after dropping some frames from the beginning and the end. The same sample size is used also in the other test set-ups.

The other data collection is performed in an office environment. The device is placed on the tripod statically and the target is moved away from the device on a track. With this workflow, the position of stations can be defined with higher precision.

In the actual analysis, only values from single central pixel of each depth map are used. Only one pixel is examined to ensure data consistency over different measurement stations. In the analysis, descriptive statistics from data series are compared, such as median, minimum, maximum, and standard deviation.

3.2.4 Depth measurement consistency over a depth map

The aim of this experiment is to examine how significant quality anomalies appear in a single depth map. Anomalies in precision are expected to follow from the way the illumination distributes to the target scene. Anomalies in absolute accuracy are expected to vary due to production tolerances and lens distortions and might thus be unique for each device. However, the intention is to examine the magnitude of these effects and not to calibrate the device.

In this experiment, the system is placed on a tripod facing directly towards a flat surface. Then, a series of depth maps is collected. The distance between the device and surface is adjusted to approximately two meters, which should be inside the effective range of the device and represent common use cases. For investigation of absolute errors, data collection is repeated in a parking garage in similar manner. The data collection is performed separately for this test, as construction tolerances of the office wall do not prove planarity of the target surface. The garage walls are concrete cast with metallic mould, which produces relatively planar surface, though some air bubbles remain and cause roughness to the surface which might disturb the measurement.

The first part of the analysis focuses on precision part of the concept of accuracy. After data collection, pixel-wise standard deviations for range measurement are computed from the depth map series. These standard deviations are used to colour corresponding pixels with a scale adjusted to the range of the values. From the produced image, real differences and possible artifacts are interpreted.

The second part of the analysis investigates areal systematic errors, those are likely produced by lens distortions. For this part, one depth map from an observation series is produced using pixel-wise median depth observations. Then, this depth map is converted to a point cloud and a plane is fit to these points using random sample consensus (RANSAC) method. Conversion from depth map to a point cloud is performed assuming pinhole camera model with camera parameters from the iPad API. The best fitting plane is searched using one thousand iterations with one centimetre threshold. Then, distances of each point to best fitting plane are computed and points are converted back to image format. Distances are illustrated with diverging colour scale and the image is interpreted to find systematic errors. The images are produced for three different data series captured from different target walls to verify that the detected artifacts are not only result of imprecise construction of the used targets.

3.2.5 Depth measurement repeatability for different materials

The aim of this experiment is to examine the effect of the target material to the accuracy of the measurement. As hypothesis, the measurement accuracy from better reflecting materials is expected to be better, whereas materials giving low intensity of returning pulse are expected to have wider spread in range measurement. This is expected as significant pulses are easier to distinguish from background noise. However, if the intensity of returning pulse is too high, it may cause sensor saturation and lead to problems for the detection.

In this experiment, the system is placed on tripod to two meters distance from the target. As the target, a 15 cm x 25 cm plastic plate is used. Thin layers of different materials are attached to the plate. In this experiment five materials are used: aluminium, cotton, paper, plastic, and wood. To be exact, the plastic target was polypropylene, and the wooden target was birch plywood. Aluminium, cotton, and wood had their natural colour, whereas other materials were white. The thickness of the target materials varies within one centimetre. However, this is accepted as the absolute range is not the main interest of this experiment.

From each material, a series of depth maps are collected from each sample in similar manner to earlier experiments. As an exception, the camera sensor is not covered during the data collection. This approach was chosen to let the system capture usable data, as the system was not otherwise capable to separate targets from the background. This finding is further discussed in the subsection 4.6. The data capture is performed in illuminated conditions.

To examine the reflected intensities, the spot pattern is observed with infrared (IR) camera. The observation is collected between capture of each depth map series, separate from depth map capture. This allows lighting to be turned off, to prevent background light from affecting intensity observation. The intensity observation is made with identical camera settings, i.e., the aperture, ISO, and shutter speed values remain the same. Shutter speed of two seconds is used to let multiple pulses aggregate the observation. Shutter delay of two seconds is used to prevent the touch of the operator to affect the result.

In the processing phase, range observations from the central pixel of depth map series are extracted and descriptive statistics are computed. The statistics include standard deviation, difference of extrema values and interquartile range (IQR). From the images of the infrared camera intensity differences between returning pulses are extracted.

To extract the returning intensities, intensity images are cropped to show only the part of the illumination pattern which hits the target. Same crop region is used for all materials. The image is smoothed with gaussian method, individual laser spots are segmented from the image and maximum intensity of the smoothed image is selected from each spot to describe the intensity of the respective spot. Smoothing is performed to average the intensity for each spot. Then, the final intensity for a material is computed as average of all segmented spots for the respective material.

In the analysis phase, the distributions of range observations are illustrated for each target material in a graph and examined visually for. In addition, the standard deviation of range observations is illustrated side-by-side with the observed intensities, and the relationship between them is interpreted. As the used infrared camera is not a measurement grade device and intensity values are not referenced to perfectly white object, intensity values are used only to compare the used materials. The data is insufficient for drawing conclusions about exact requirements for target materials.

3.3 Software and analysis

In this work, an app called *Stray scanner* is used to capture depth maps, corresponding confidence maps, RGB video and camera parameters with the examined tablet device. Finally, data processing was done using Python in Jupyter Notebook with help of some common data processing libraries. These include Numpy (Harris et al., 2020), Open3d (Zhou et al., 2018), OpenCV (Bradski, 2000) and SciPy. Most data processing is performed with Numpy. Some more complex or specific tasks are executed using SciPy and OpenCV. These include interpolation and smoothing, for example. Open3d is used to convert data from depth map format to point cloud, when necessary, and for handling the data in that format. All graphs are plotted using Matplotlib plotting library. Some point cloud representations have been created with Open3d. The complete processing notebook is presented in the appendix of this work.

4 Results and discussion

This section is divided into subparts following the same pattern as in part 3.2. The subparts reveal the results of corresponding experiments and discuss their quality and meaning.

Throughout this section, several of the results are illustrated with figures. Each time a figure displaying important results is shown, the text explains choices in visualization, highlights important features in the figure and then continues to the discussion of these characteristics.

4.1 Illumination unit pattern and intensity distribution

This subsection describes the results obtained with methods described in the subsection 3.2.1. The result of the first test, which aimed to find out the distribution of illumination power to target scene and pattern of laser light, is presented in figure 10. The outcome of the other test is illustrated in figure 13. The aim of the latter picture is to verify the effect of illumination direction, i.e., whether the illumination power of laser unit is dependent on the direction, or are anomalies seen in a single image caused by the reflective properties of the object.

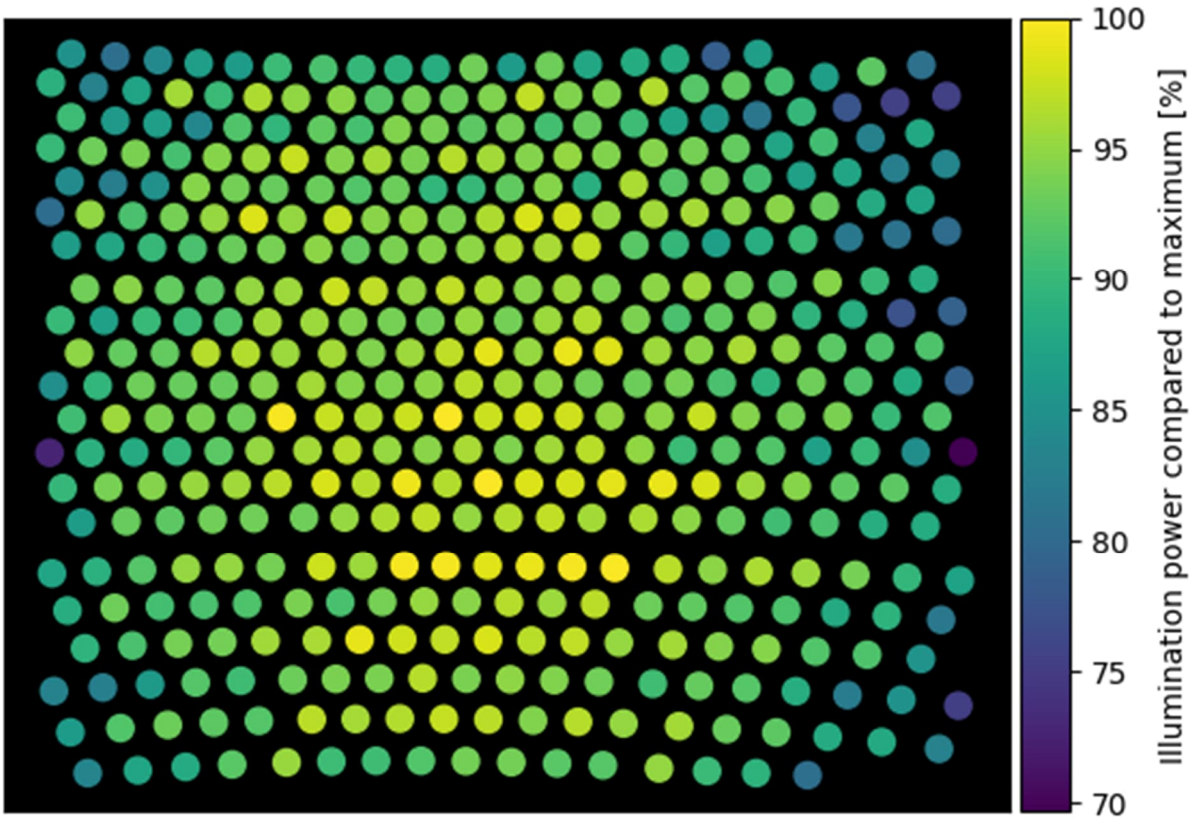


Figure 10. Illustration of intensity scattering back from the target wall. Individual spots have been segmented.

In figure 10, each marker represents a single laser spot. The color of each marker represents the intensity of the corresponding spot. The colour scale is illustrated in the right part

of the image. The markers are plot as seen from the perspective of the IR camera, and thus the pattern is distorted in the vertical direction.

In the figure, the repeating pattern of laser dots, size 8x8, can be noticed, though all individual replicas do not fit into the field of view of the IR camera. The pattern is further highlighted in figure 11. The whole illumination matrix contains nine of these smaller patterns. In the picture the colour of each dot describes the fractional intensity reflected from a dot compared to the highest intensity dot. Looking at the picture, the intensity seems to drop when moving from the centre towards the edges of the pattern.

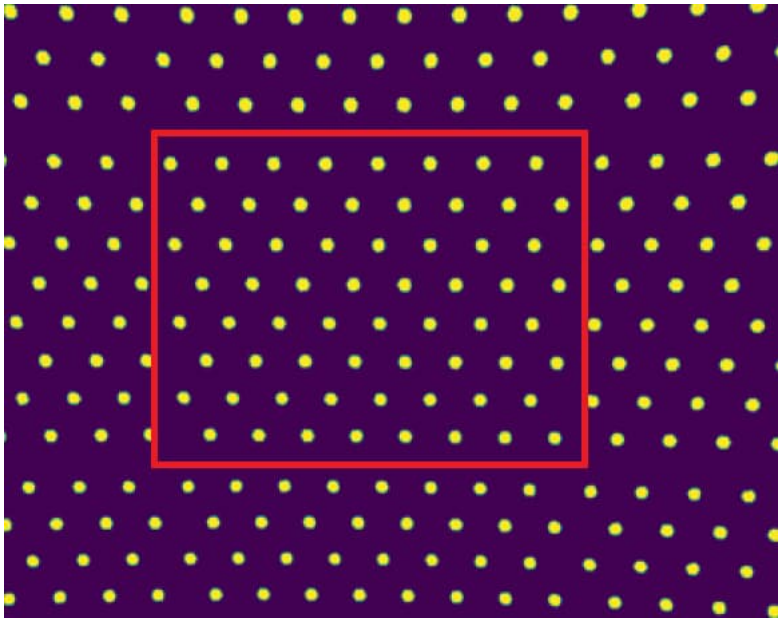


Figure 11. One tile of the complete tesseling pattern, highlighted with a red rectangle

Next, the extracted dots are interpolated to a continuous field to help examination of continuous changes in the reflected illumination power. The result of the interpolation is present in figure 13. However, first, figure 12 presents an illustration of the captured data. The aim of the representation is to clarify the experimental set-up used in the measurement.

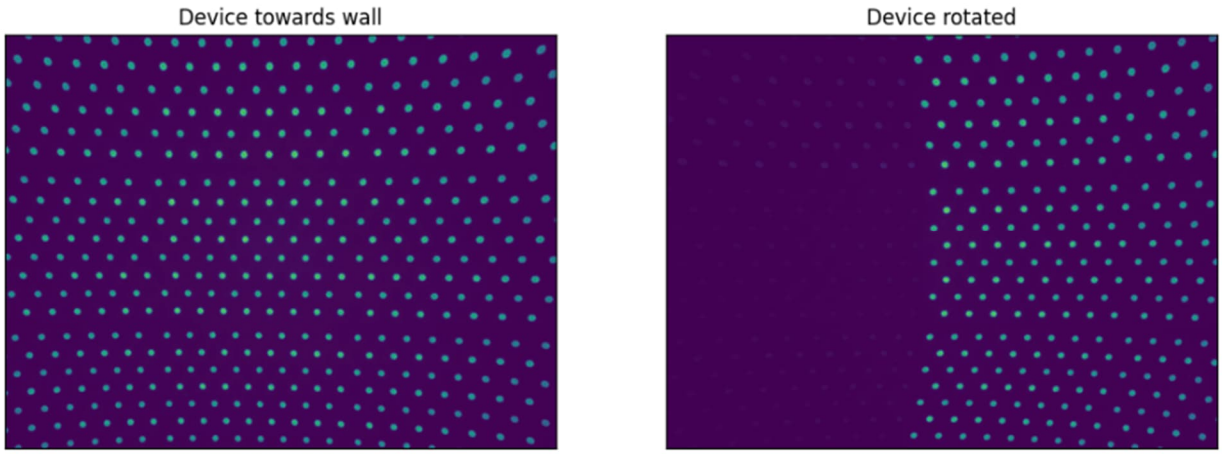


Figure 12. A clarifying illustration showing the data used in the interpolation.

Figure 12 shows side by side data extracted from the IR camera in two experimental set-ups. In the first picture, the device was turned directly towards the target wall, and illuminated laser spots cover the whole field of view, whereas in the other picture the device is panned, and the centre of the image covers the left edge of the laser pattern. In the image, bright points correspond laser dots. In the image where the device is rotated some reflections are present from the surrounding walls.

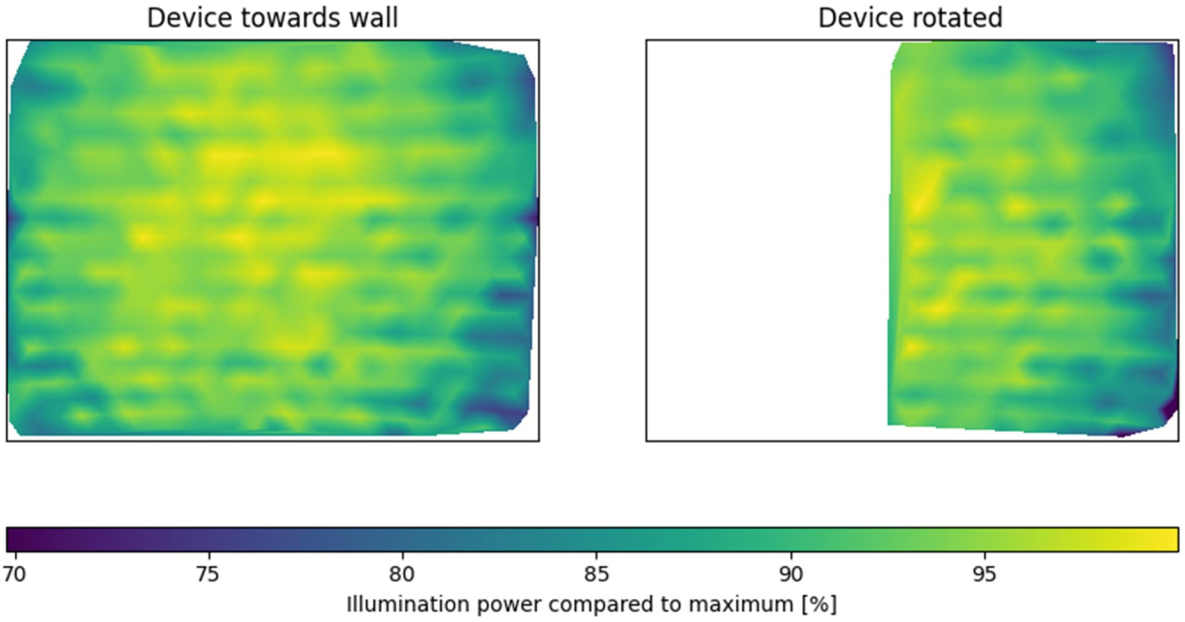


Figure 13. Comparison of interpolated intensity fields. In the first case (left) the system points directly to the wall, whereas in the other the LiDAR device is panned.

In figure 13, the left part shows intensity field interpolated from laser dots when device is facing directly towards the wall. Correspondingly, the right part illustrates the intensity field when the tablet is panned right. Now, in these both images, the highest intensity dots are in the centre of the image. The highest measured intensity level is napproximately the same in both cases. In the first case this represents the centre of the light pattern, whereas in the other case, edge spots are in the middle of the picture. In both cases the intensity decreases when moving away from the centre of the image.

From the previous information it can be deduced that the decrease in intensity seen in figure 10 is not a result of uneven power distribution in the illumination unit. Instead, the decrease towards light pattern borders follows from larger footprint of a light beam, and reflective properties of the object. Reflective properties are important, as equally bright response to different viewing angles would require a completely diffuse, Lambertian surface, which rarely exists in the nature (Westin et al., 2004). When reflective properties are such that big proportion of photons are reflected away from the sensor, ranging becomes more difficult as signal-to-noise -ratio is smaller.

During the experiment, also the field of view of the depth sensor was checked. The correspondence to the opening angle of the illumination pattern was examined. The measurements and the result are represented in the table 1.

Table 1. Summary of FOV definition

	Property	Value	Result
Point pattern opening angle:	**Distance to the target wall**	175 cm	~ 59.0° x 45.8°
	Point pattern horizontal width	198 cm	
	Point pattern vertical width	148 cm	
Sensor FOV:	**Focal length**	1625 px	~ 61.1° x 47.8°
	Depth map horizontal width	256 px	
	Depth map vertical width	192 px	
	Conversion scale	256 px / 1920 px	

The table lists the measured quantities and values, together with the calculated result of opening angles. The result shows that the sensor observes slightly wider view than the point pattern illuminates, when considering the central axis of the image. However, areas in the scene corresponding to the corner areas of the sensor are covered due to the distortion of pattern in the corners. Nevertheless, as the scene is illuminated with a point pattern instead of continuous beam, unilluminated parts in the scene are natural for the approach. Areas without light pattern cover exist even if the pattern fully extends from edge to edge of the sensor. Thus, the difference between these angles is not a serious issue.

4.2 Importance of RGB image in depth map creation

This subsection describes results obtained from experimental set-up of the subsection 3.2.2. The resulting comparison of produced depth maps with and without the impact of the RGB colour image is presented in figure 14.

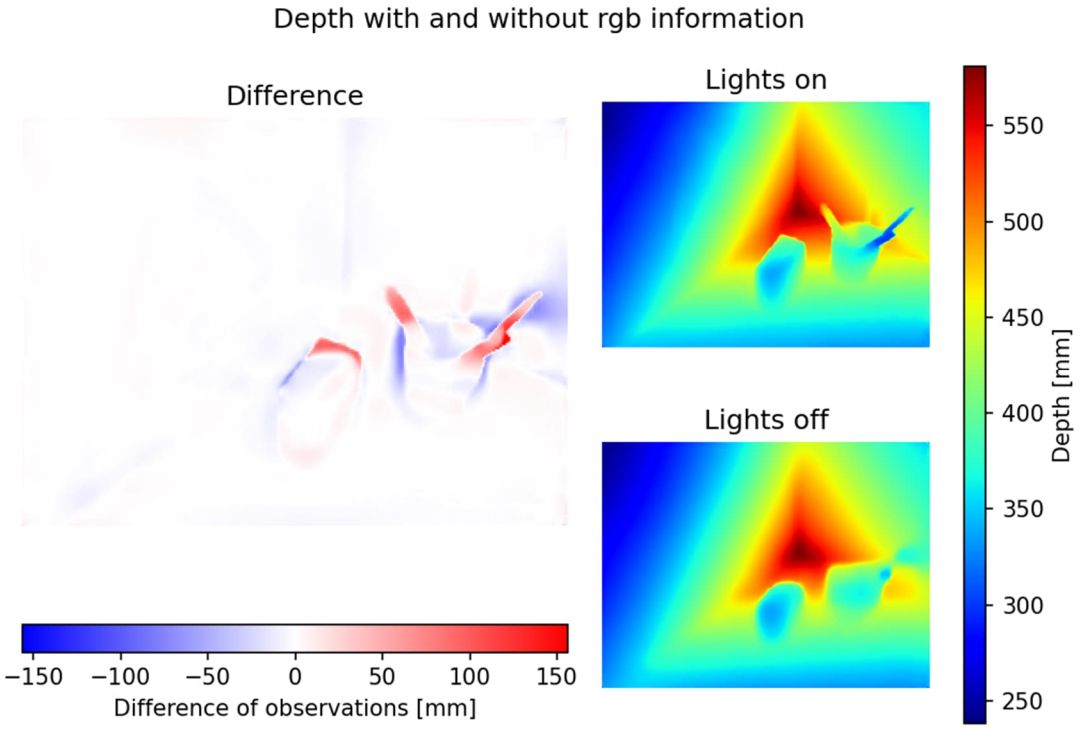


Figure 14. Comparison of the depth maps collected with and without RGB information.

The figure includes depth maps created from median values of the observation series on the right. To help interpretation, the difference of them is present in the left part. The difference is calculated by subtracting the depth maps from each other. In the visualization of the difference, red colour represents areas those the system observed to be closer to the camera, when the assisting RGB image was in use.

Largest differences in the picture appear in border areas of each object, in the transition area between object and background. Some differences also appear in the room corner in the background of the scene. The picture of the geometrical object does not make any visible response to the difference of the images. In the depth map, which is captured without image information, the cup of pencils seems to blend into the wall on the right, as those are close in same depth. In general, the depth map produced without supporting image information seems like a smoothed version of the other one, also missing many small details.

When comparing these two cases, it seems that the use of image information improves evidently the quality of depth information by adding details and refining corners. This is best seen in the difference image, looking at the highlighting pencils and the block of notes. Some pencils that clearly show up in the image assisted depth map are totally missing in the first depth map. The edge of the note block is also much sharper when image is used to help the depth map creation. In addition, the edges of the corner seem to sharpen slightly, when looking at the wall colours carefully. Nevertheless, when the depth data is viewed from different perspective, some shortages can be found in the improvements of detailed areas, as figure 15 reveals.

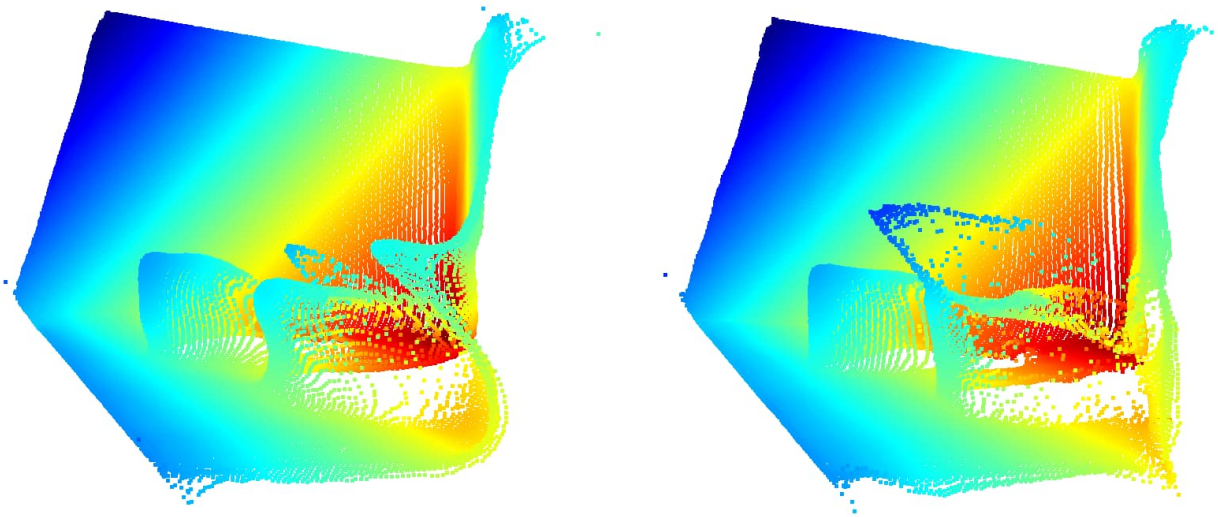


Figure 15. Point cloud representation of generated depth maps. The data capture displayed on the right was assisted with image information.

At first sight figure 15 seems to verify remarks pointed out in the previous paragraph. The wall edge on the foreground repeats much better, and whereas some pencil is a strange bump from the background in the first case, with image information it seems to be an object. However, when looking at the pencils carefully in the right-hand side point cloud, it shows that these small details blend to each other. When looking at figure 8 and figure 14, it seems that the two rightmost pencils are well repeated with the help of image information. Although

these two pencils point in different directions, the first towards the sensor and the other away from it, only the first is well repeated in figure 15. Depth values corresponding the other pencil seem to extend from the tip of the first pencil to a one depth observation received from the other, rather than representing any real object.

From this behaviour we can draw a conclusion, that the depth map creation does not take advantage from stereo imaging possibility. As the device includes two camera sensors, it could use stereo image processing techniques for depth map improvement, but as the depths can differ this much from the actual depth values, it is likely that the image helps the process by helping to segment objects from the scene. However, data collection applications storing the data could advantage from filtering with range reliability as described by Weinmann (2016, 28–29). The idea of the method is to leave out observations from a depth map in areas where standard deviation within adjacent pixels is high. This would most likely remove significant number of false points from the edges of objects, those currently presented to lie between the background and the foreground object.

4.3 Range measurement repeatability

This subsection describes the results obtained in experimental set-up of the subsection 3.2.3. First, results from longer range experiment are present. After these, corresponding and supplementary information from short range test are presented.

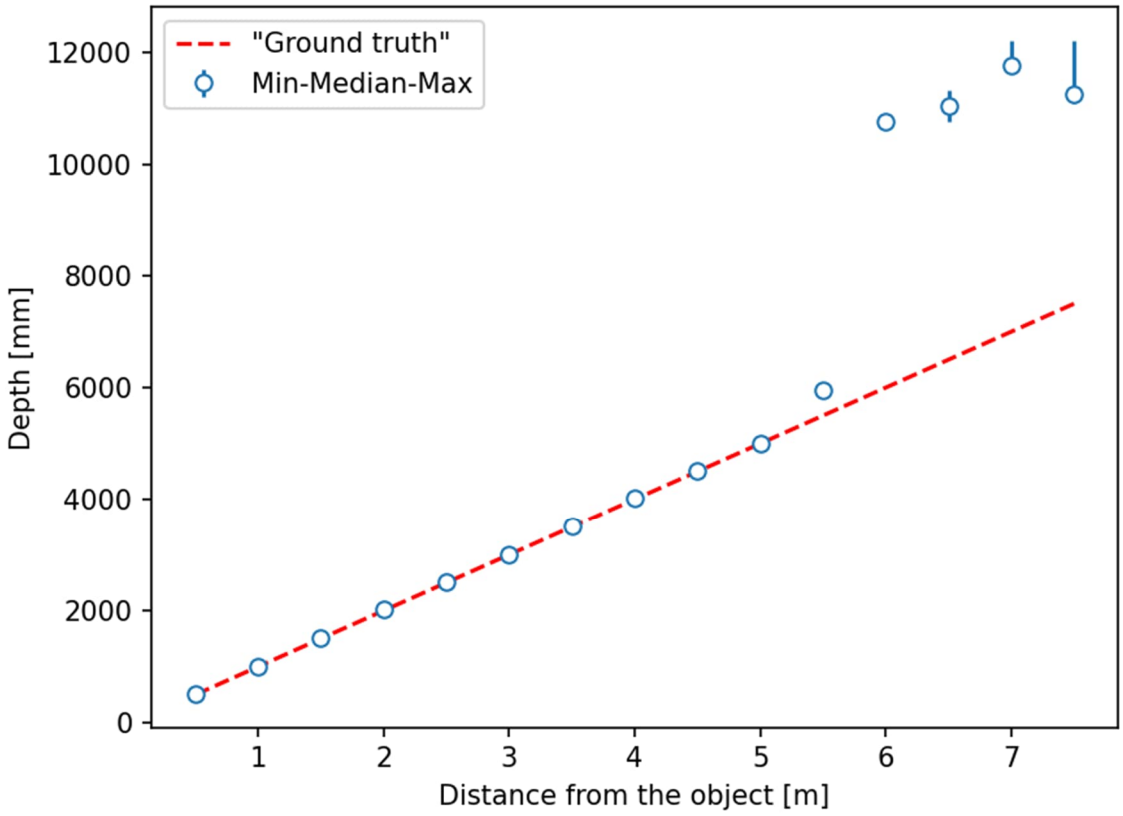


Figure 16. Overall representation of the results obtained in the parking garage.

Figure 16 shows an overview of the data collected from different ranges in the parking garage. The horizontal axis shows each measurement station whereas vertical axis plots observed range values. Observed median values from each station are expressed with blue circles. Vertical lines represent the interval to minimum and maximum values on stations where it is wide enough to not hide "behind" the median marker. A red dashed line represents the "ground truth" of range values, where median markers should lie in ideal situation. If markers are located somewhere else, either measurements have error or placement of measurement stations is imprecise.

As seen in the figure, the overall quality of the data seems good as most of the markers are located on the reference line and spread seems to be rather low. However, when measuring targets are located further than five meters away, the measurement seems to suffer from significant loss in accuracy. This realises as significant systematic error in measurements and increased spread. For example, the measurement collected from station that is six meters away from the object, the observed range is 11 meters meaning five meters systematic error. The system overestimates the distance to the target. Even though the placement of measurement stations is slightly inaccurate with tape measure and directing the device to the target by hand, inaccuracy of this magnitude must follow from system related issues.

The found systematic error might be caused by various reasons. It might be related to limitations of the technology or problems in processing of the data. One possible limitation of the technology leading to range overestimation is multipath problem. In this situation the emitted laser light reflects from multiple surfaces before it returns to the sensor. As the sensor is not capable of separating rays that return directly from the target and others that have travelled longer path, the system may overestimate the range. However, during the planning of these experiments, similar large systematic errors were observed but to opposite direction. As the same system can produce systematic error to different directions, both under- and overestimating the reality, the multipath problem cannot be the only explanation, as it can at most explain only part of the problems. Data collection for initial testing was performed in different environment.

A processing solution that might cause this behaviour is the use of image data and LiDAR sensor data together to produce the depth map. As LiDAR data and image information are used together to build the depth map, it is possible that the system is disturbed in this test when the camera is covered and provides constant black feed. It is possible that system is configured to rely more on the image information when the LiDAR data becomes less reliable.

Next, the spread of range measurements is analysed. From the previously shown data sample, observations collected in stations from half meter to five meters are used. Samples collected from longer than five-meter range are neglected as the data seems to be too disturbed according to previous experiment. The statistics computed from the data are present in table 2, and a visual representation of standard deviations is available in figure 17.

Table 2. Standard deviation, width of interquartile range and min-max interval of observations from each station.

Station	σ [mm]	IQR [mm]	min-max [mm]
0.5 m	0.54	1	5
1m	1.11	0	16
1.5 m	0.68	0	11
2 m	1.22	0	17
2.5 m	0.97	0	8
3 m	1.07	0.5	10
3.5 m	1.11	2	11
4 m	2.11	4	15
4.5 m	1.63	3	19
5 m	1.75	0	12

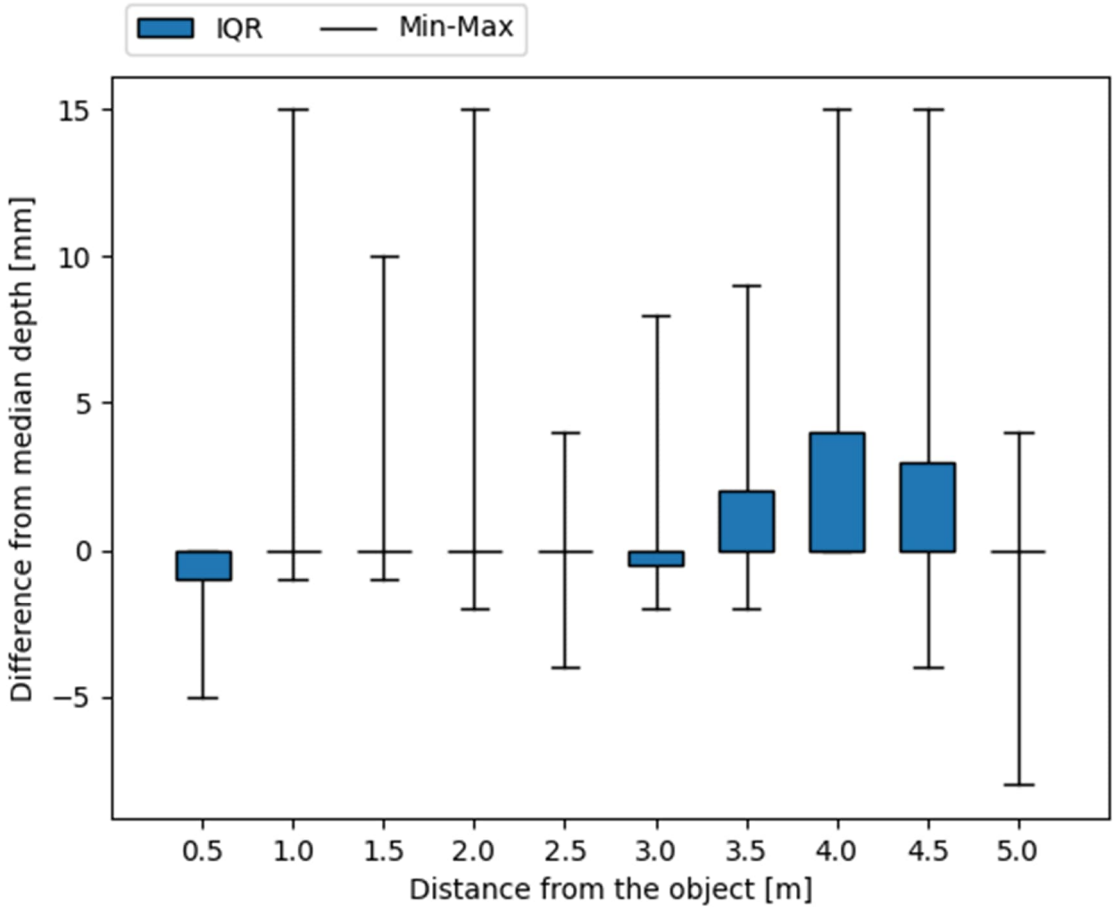


Figure 17. Spread of the range measurement from half a meter to five-meter distances.

The figure shows how widely range measurements of the device spread around the median value. Blue bars illustrate IQR, which covers at least fifty percent of all observations from each series. In addition to that, complete interval of observations, extending from minimum to maximum at each station, is shown with whiskers. Values are plotted relative to median of observations at each station.

Interestingly, range observations that differ from the median seem to be more likely larger than the median value. This emerges as extreme value spans extending further to positive than negative values. The most probable explanation to this effect is the timing solution Apple has used for the system. A patented solution, which is likely used in the device, first roughly defines the distance to the target, then improves accuracy by building a histogram of first photon arrivals within that interval and finally deduces the range from the histogram (Sharma et al., 2020). The use of first photon arrivals most likely causes the found bias.

Characteristics present in the figure do not show a strong relation between spread and distance within the range of five meters. The difference of minimum and maximum seems to remain approximately the same. The IQR does not systematically grow but over half of observations are captured to same depth on five-meter range as well as from one to two and a half meter. However, when looking at the standard deviation values, which is presented in figure 18, a slight increasing trend in the deviation is present.

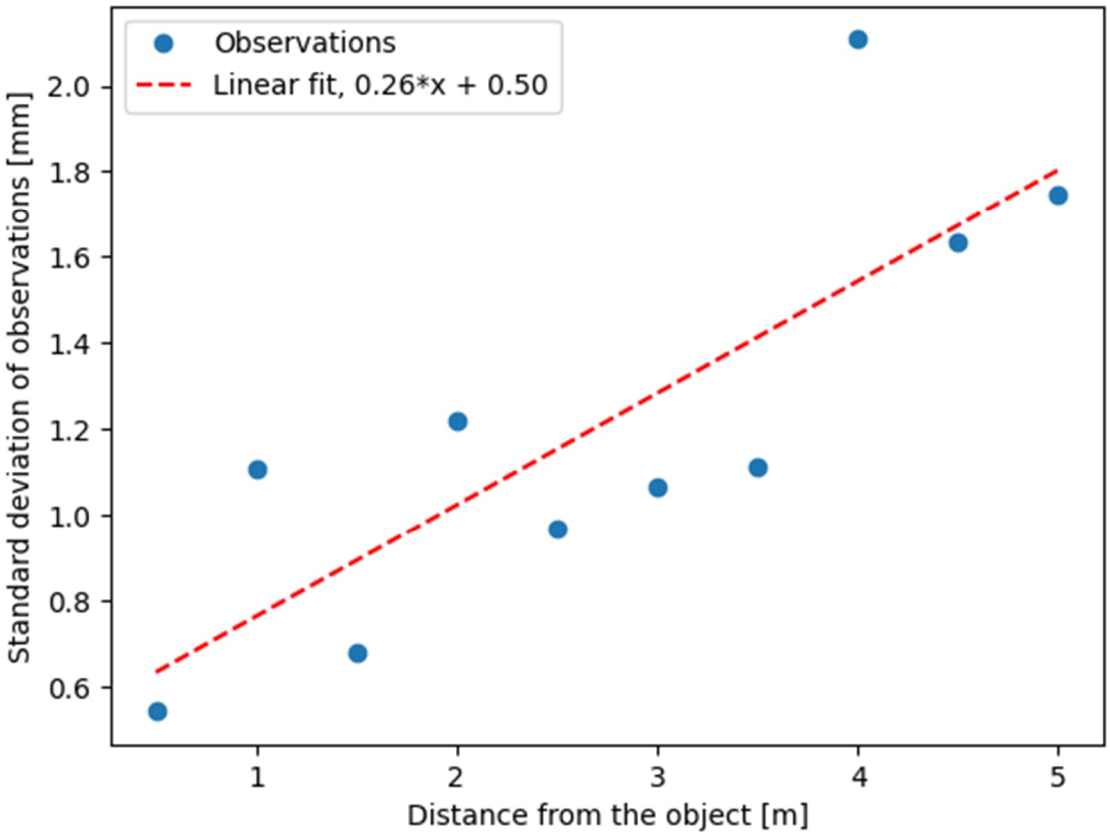


Figure 18. Relation between observation distance and standard deviation.

In the figure, blue markers represent standard deviations from observation series collected from each station. The red dashed line is a linear function fit to the data with least squares method. The equation for the function is presented in the legend.

Compared to the strong linear relation that Giancola et al. (2018) found analysing Microsoft Kinect V2 sensor in corresponding manner, the correlation is quite weak. The

coefficient of determination, R^2 they found for the linear correlation was 0.9916, whereas here the corresponding value is only 0.6642. However, looking at the parameters of the linear fit, iPad sensor seems to gain less noise when the distance to the target increases. Giancola et al. (2018) measured 0.78 mm/m, whereas here the same value is 0.26 mm/m. Giancola et al. (2018) found linear correlation to begin from 1.5 m distance. At that distance, the standard deviation was about 1.2 mm, which is about half a millimetre more than the fit shows for the iPad.

When evaluating the depth measurement standard deviation of the system, it is important to note that the system does not provide access to raw flash LiDAR data, but a depth map is always product fused from images and depth sensor (Apple Inc., 2020b). According to the experiments discussed here, the system for example seems to smooth depth images, which should reduce the level of noise. Also, the angle between the system and the target surface seems to affect how the noise appears throughout a depth map, which may explain the noisiness of the observed standard deviations displayed in the figure. This behaviour is introduced in the following subsection. Nevertheless, first, the results from measurements made on short range are presented in figure 19 are discussed.

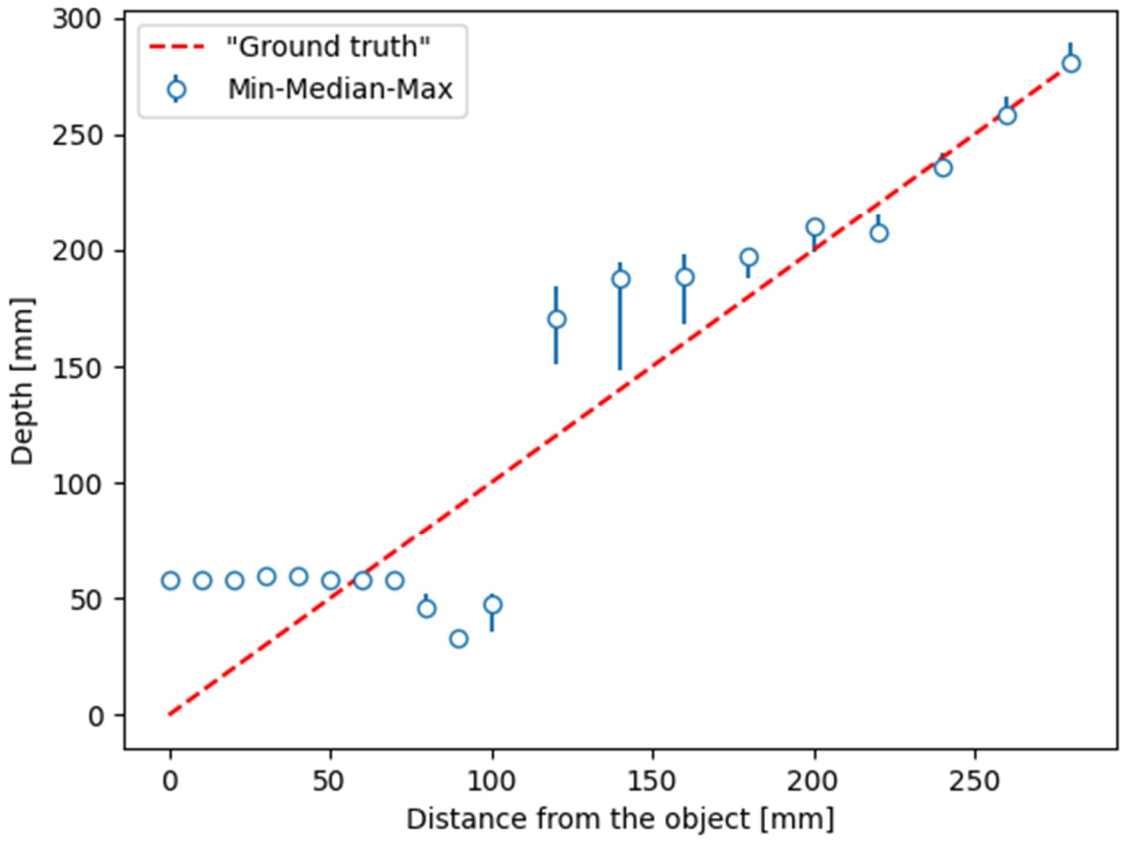


Figure 19. Overview of measurement spread on short ranges.

The figure above illustrates how measurement noise behaves when collecting data with the sensor from short ranges. As in figure 16, observed median depths are indicated with

circle markers, intervals between observed extrema values are illustrated with blue lines and the expected ground truth is highlighted with the red dashed line.

Looking at the plot, the device seems to produce nearly constant observations until the target is further than 70 mm away. Then, the system drastically underestimated the range, until the distance exceeded 100 mm and the system started to overestimate the range. After the target was moved further than 200 mm away, observations began to be in line with the reality. Notably, in the range from hundred to two hundred millimetres, the spans between extreme values nearly reach the true value, meaning that the system every now and then produces valid observations, though the median was significantly off the truth.

Considering this behaviour of notable systematic error disturbing the measurements, it is reasonable to state that the effective range of the system starts from 20 cm. In addition to that, it seems to end at 5 m, according to same effect in the other end of the range presented in figure 16.

4.4 Depth measurement consistency over a depth map

In this subsection, depth map data is analysed as a whole, whereas in the earlier subsection the analysis extracted only single pixels from each depth image. first, the deviation of depth values for each pixel are analysed. An illustration of the data is presented in figure 20.

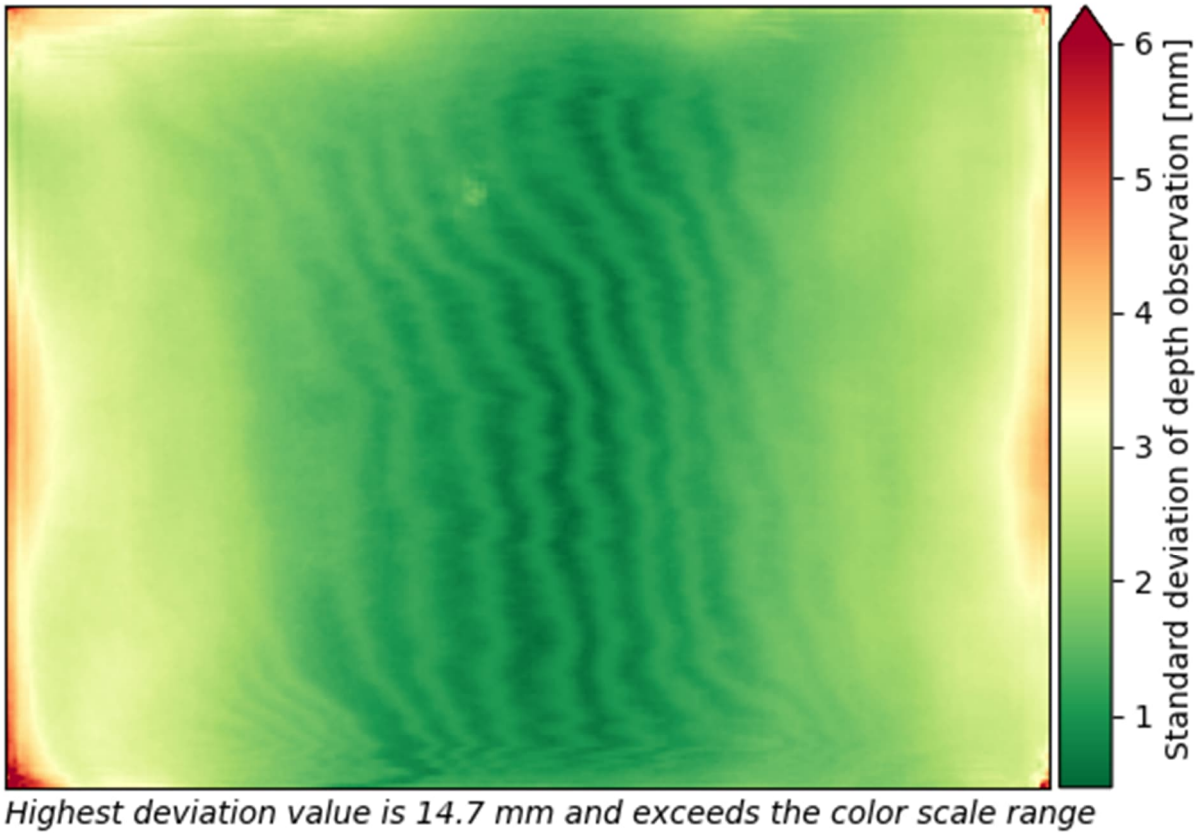


Figure 20. Standard deviation of pixel depths over the sensor.

The figure shows in image format how the spread behaves over a depth map. The colour scale on the right presents what value each colour tone corresponds. Green tones represent lower deviations, whereas the spread is wider in red pixels.

In the image, the deviation for central pixels seems to be smallest. This is expected, as in section 4.1 the intensity of returning light was found to decrease towards borders and it has direct impact to the quality of detection. Near edges, and especially in corners of the image, noise increases rapidly. This may be affected by some additional property than only decrease of returning light. The changes may, e.g., be result from the combination of flash LiDAR depth data and image information. As the system illuminates light in a spot pattern rather than continuous wide beam, sensor might end up having buffer of edge pixels without any returning light and the system must extrapolate the depth data there using only the RGB-image, which is not as reliable as LiDAR method.

In the central part of the image, changes in deviation appear in wave pattern, making vertical stripes to the image. These follow from the limited depth resolution of the system. The system produces depth maps storing the depth value as an integer number in millimetres. Thus, pixels covering target in even millimetre depth become more reliable than pixels where the true depth is between two values. This behaviour makes the orientation and placement of the sensor important. In the previous subsection, coefficient of determination between observation distance and measurement spread was found relatively weak, and observations spread around the fitted line. Now, the found behaviour may explain the weak correlation. It is possible, that in some measurement stations the real depth of the central pixel had larger fractional part. Thus, the error magnitude of depth observations varied due to this and the standard deviations spread from the fitted line.

Next, the text introduces systematic anomalies in depth map, those affect the absolute accuracy. First, overviews of the data samples are introduced. After that, the actual results and discussion will follow. The overview of the two collected data samples is shown in figure 21.

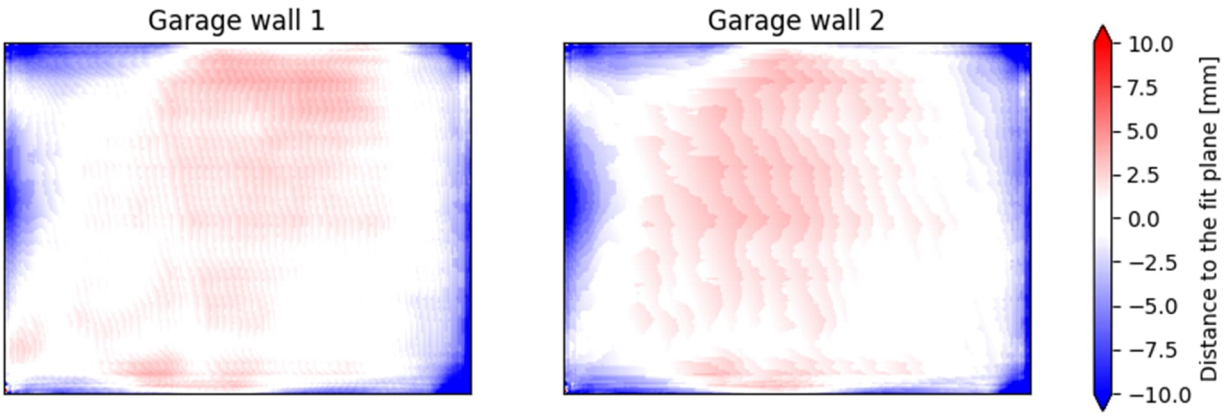


Figure 21. An overview of data samples collected from two different target walls

The figure shows distances of points to the fit plane in diverging colour scale. Red pixels are closer to the camera, whereas blue points are behind the plane when looking from the camera direction. All pixels closer than 1 cm to the fit plane are completely white, and the tone starts diverging when the absolute distance exceeds that limit.

The aim of displaying these two datasets side by side is to point out artifacts in the data, that are not actually result from system errors but present in the target wall. Looking at these two illustrations, figures remain mainly the same. Two important differences are the position of the red area in the centre, and distribution of the wave pattern. The red pixels are slightly closer to left edge of the second picture. The wave pattern in the first image has higher frequency, as the angle between the device and the wall has been slightly larger during the data collection.

For further analysis, these two depth maps are fused to one by averaging distance values between the two alternatives. The result is also smoothed with gaussian kernel of nine pixels to reduce small artifacts and highlight bigger phenomena. The result is present in figure 22.

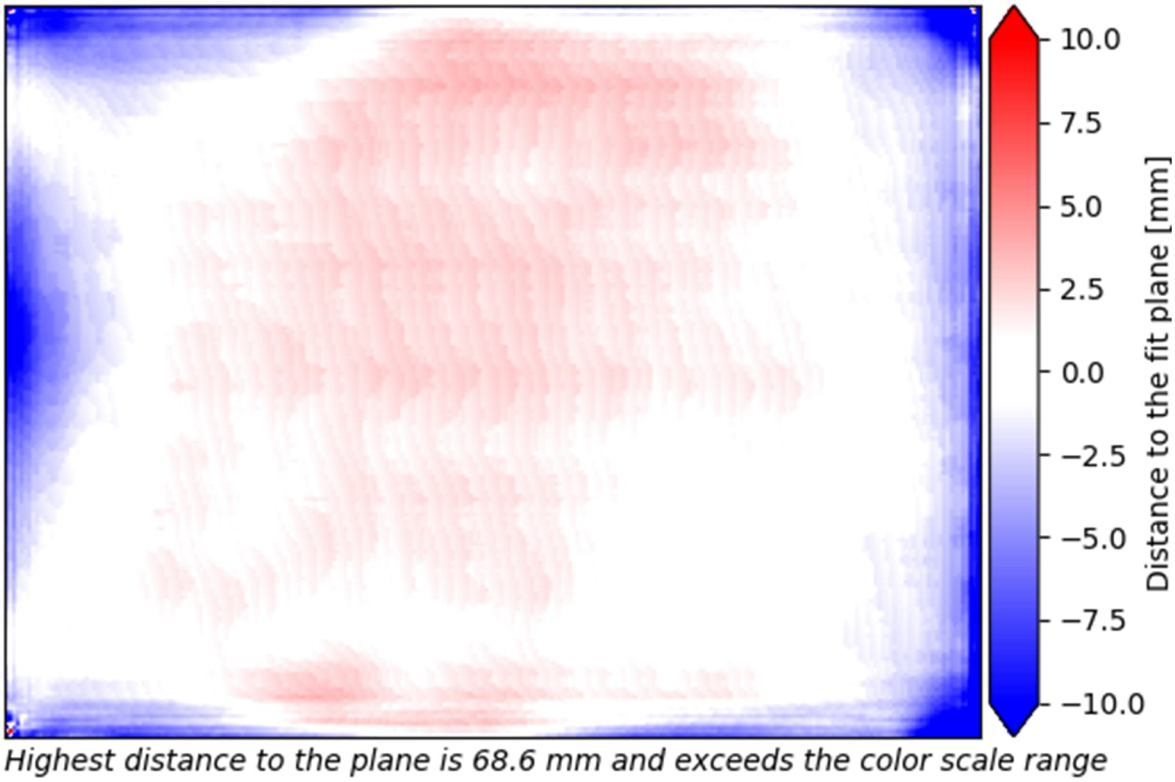


Figure 22. Averaged and smoothed distances from plane fit

Like in previous visualization, blue pixels again denote locations behind the fit plane, and red pixels points closer to the camera. In general, blue areas are focused on the edges and corners of the image, whereas red pixels are in the centre of the image.

From this data, it is impossible to say which parts of the depth map are detected to the correct depth. Instead, the system clearly measures planar surface as convex. This may reveal the system suffering from barrel distortion. In case of barrel distortion, edges of the image pack closer to the image centre compared to ideal pinhole camera model, and in this context target surface further away are captured to the image edge (Giancola et al., 2018, 11). When compared to a plane, the data creates a convex effect. However, this distortion effect may be also result from other production tolerances than optical distortions but as investigating the level of distortions and their absolute sources is not the major aim of this work, this is not examined further. It could be, that the seen effect is a result from image processing methods that have problems in the edges of the image. Nevertheless, for this data, the

possibility of that plane curvature really is present in the target due to construction tolerances cannot be neglected.

4.5 Depth measurement repeatability from different materials

In this experiment, the effect of the target material is examined in the manner introduced in subsection 3.2.5. During the experiment, intensities of reflecting pulses were monitored with the help of an infrared camera. The values representing the intensity of returning pulse describe the intensity as fraction of the effective range of the infrared camera. The numbers should only be used to look for relations between materials within the experiment. All materials, observed intensities, and corresponding accuracy statistics are presented in table 3.

Table 3. Summary of accuracy characteristics obtained for different materials

Material	Intensity of reflected pulse [-]	σ [mm]	IQR [mm]	min-max [mm]
Aluminium	0.86	1.23	2	9
Cardboard	0.24	0.99	2	6
Cotton	0.26	0.72	1	4
Paper	0.22	0.90	1	6
Plastic	0.17	0.90	1	7
Wood	0.24	0.70	1	5

In the table, target materials used in the experiment are listed. For each material, the intensity of returning pulse, together with the standard deviation, IQR, and extrema range of the range measurement are presented. From the values, it is important to note that the intensity observed for aluminium is not completely comparable with other materials. This is influenced by two reasons discussed in the following paragraph with the help of figure 23.

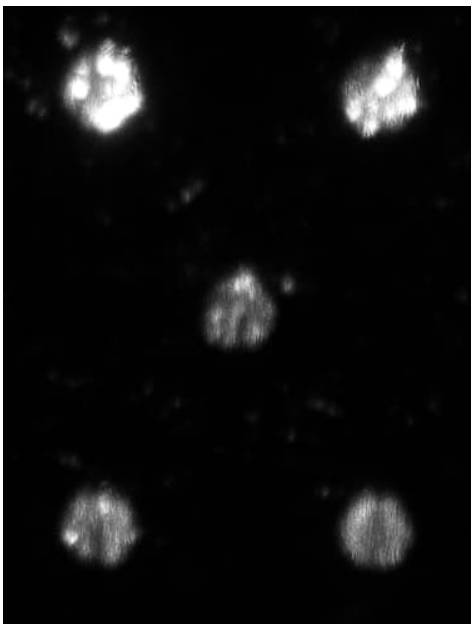


Figure 23. Reflection from aluminium target with some saturated pixels

The figure above shows the pattern of five laser dots, which projected to the target plane during the experiment. However, from aluminium target, some laser dots mirrored to the infrared camera leading to pixel saturation. In other words, the sensor was not capable to measure all the photons, and the value for measured intensity in the table should be higher. Nevertheless, spots that did not cause saturation, but the reflection was rather diffuse, would have resulted approximately to value 0.5 in intensity. This value is smaller than the one presented in the table. However, the difference between these two options is not important from the analysis part, as analysis is performed visually, and the intensity is anyhow higher for aluminium than for any other material used in the experiment. Next, the standard deviations in the table are illustrated in figure 24.

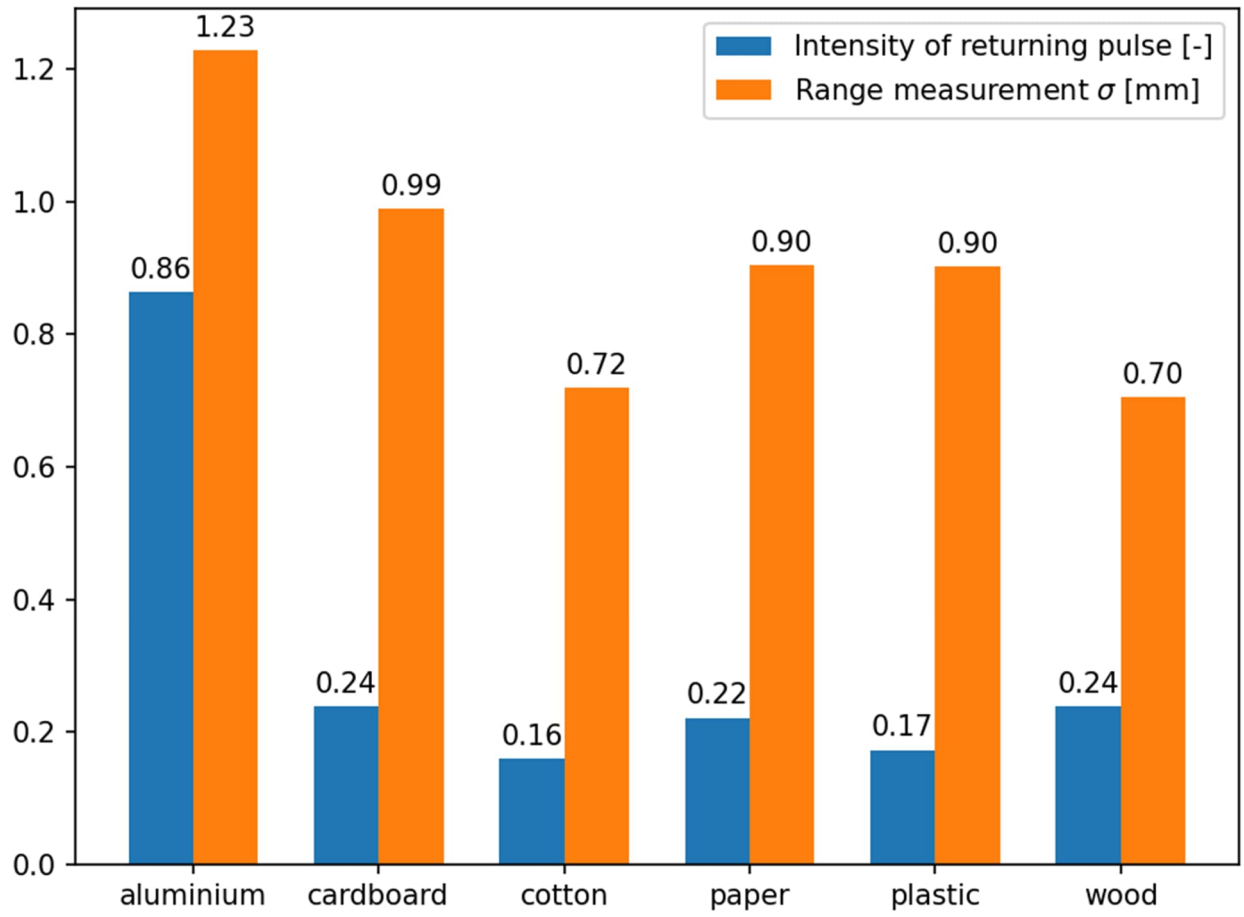


Figure 24. Illustration of the relationship between reflectance of a material and the standard deviation of the range measurement

The figure shows for each target material the measured intensity of the returning pulse, and the standard deviation range measurement. Values are illustrated with the height of blue and orange bars, respectively.

Both the highest range variation and highest intensity are observed for the aluminium target. Lowest intensities are measured for cotton and plastic targets. Standard deviations

for wood and cotton targets are the smallest. Nevertheless, except from aluminium target, all the statistics seem to be nearly the same for all materials.

Surprisingly, these observations do not show any proper relation between target reflectivity and measurement accuracy. Only the accuracy from the aluminium target is lower than from others, which probably is caused by problems following from reflecting surface. The difference is relatively small though. The data processing solution developed by the Apple might help the system in problematic places like this with interpolation and filtering. Nevertheless, it is possible that some relation may rise if the system is pushed on its limits. For example, if the target were further away, leading to lower amount of light to return to the sensor, the strength of the returning pulse might not anymore be enough for detection of the returning pulse.

In addition to statistics, bare distributions of observations are illustrated and interpreted in this work. The analysis is present in the next paragraph following figure 25, which illustrates range measurement distributions from the target materials.

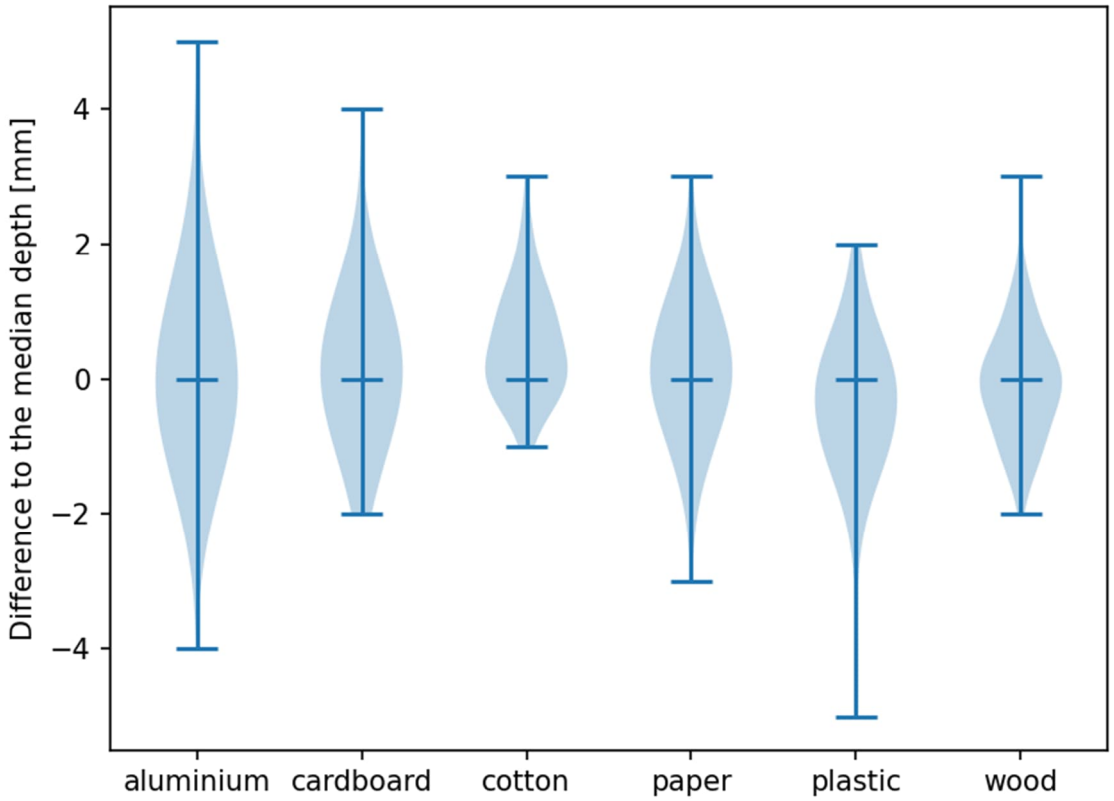


Figure 25. Violin plot of depth measurement distributions from targets of different material

The figure above shows the depth measurement distributions from all used target materials in a violin plot. The width of a blue band illustrates the occurrence of observations on that distance. All distributions are plotted around their median value, so only the shape of each distribution matters and comparison of absolute accuracy is not possible. Absolute differences are not shown, as perfect placement of target on the same place after the change of material was not possible in this test set up. However, the data was checked before plotting, and notable differences in median distances were not present, but the median value deviated within half a centimetre for all materials, which corresponds well with the difference of

thickness of target plate and uncertainty from placement of the target. In addition, minimum, median, and maximum values are highlighted with blue horizontal lines for each material.

The visualization shows the spread of depth observations being largest for aluminium, and smallest for cotton. Otherwise, the shape of distributions seems close the same for other targets, though the width between extrema values deviates from sample to sample. The distribution of measurements from cotton targets seems to be slightly biased to include more larger distances, whereas other distributions spread more equally around the median.

The reason observations from cotton are slightly biased might come from the surface properties. Whereas other materials in the test are solid, cotton textile is porous and may let the light pulse partly transmit through it. Then, the light may reflect from lower layers of the textile or from the plastic plate the material is attached to. If returning pulse is identified inside the material, the range observation will be slightly larger, and result as bias in the distribution. For the wider spread of the distribution from aluminium target the discussion stated with figure 24 applies.

In the end, based on the collected data, it is hard to point out dependency between material reflectivity and measurement repeatability. Except the issues described throughout this subsection, the differences in accuracy statistics and measurement distributions seemed small. These small differences may be related also to something else than physical properties of the target, such as the target orientation related accuracy dependency discussed in the subsection 4.4.

4.6 Summary of findings

In this work, multiple experiments were made, aiming to examine the accuracy of the flash LiDAR sensor packed in the Apple iPad Pro (2020). These experiments included examination of the illumination unit, visual interpretation of the use of colour images in depth map creation, relationship between error magnitude and observation distance as well as target material reflectivity, and interpretation of data consistency over a depth map. In addition, notes about characteristic features of the system were made throughout the work. table 4 below summarizes these features, which are important to note when considering the use of the device in practical applications.

Table 4. Summary of iPad Pro (2020) depth sensing characteristics

Property	Value
Depth map resolution	256 x 192
Field of view	~ 61° x 47°
Operative measure range	from 20 cm to 5 m
Depth resolution	1 mm
Frame rate	60 fps
Confidence map values	3

One significant aspect affecting the measurement accuracy of the depth sensor is the solution it uses to combine depth observations and colour images. This system prevents the use of raw depth observations, as software developers are currently provided only the

processed data. The system seems to apply significant smoothing on the depth maps, which is illustrated in figure 26.

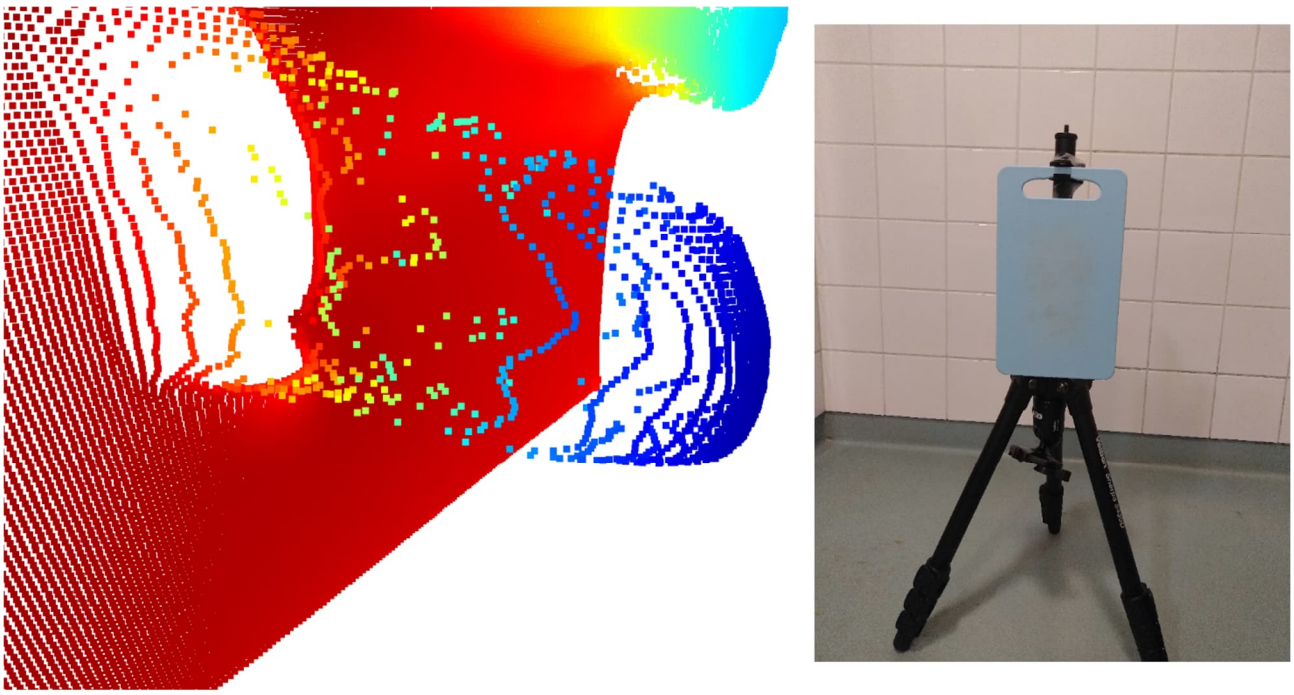


Figure 26. Illustration of captured points from the side perspective. Data was captured from 2 m distance, camera covered. The picture on the right illustrates the target

The left part of the figure shows a point cloud representation of a depth map captured during the planning phase of the data capture from targets of different material. The right part illustrates the 15 cm x 25 cm target plate from the front, whereas the left part shows the point cloud from side. During this capture, the camera sensor was covered. When the target was in front of the background, the system resulted in very smooth depth map observation when the colour image feed was not helping to segment the target from the scene. Thus, the system should not be used in general case without the assistance of camera information.

It is worth to remember that the system is designed to support AR applications, and not measurement purposes. In AR applications, the smoothness and continuity of the data flow override the need for detection of all small details, as those provide more fluent user experience. Probably thus, the system is also built to perform 60 frames per second.

Nevertheless, the system seems to provide rather accurate data within its effective range. As the data may capture some artifacts, cleaning of the data is necessary if the data itself is used as a product. For example, cleaning each depth map with range reliability metric described by Weinmann (2016) could help to remove many error points from the data. The metric evaluates the reliability of each depth map pixel by the variation of depth values within a small neighbourhood, thus helping to remove observations located incorrectly between back- and foreground.

5 Conclusions

The aim of this work is to discover how accurate information can be collected with the depth sensing system of Apple devices and characterize the system from metrological point of view. The information should help the assessment of possible applications. Though the depth sensing system has many different technologies integrated together with the actual depth sensing flash LiDAR, the investigation focused on the performance of the flash LiDAR and tried to neglect the other parts of the system. The reason behind this decision was an assumption that other data sources are mainly used to support and complete the observations from the depth sensor, while the depth sensor observations provide the base for data feed. In addition, earlier studies had already been made assessing the data usability of the complete system, but investigation of depth data separately was missing.

The literature review part of the work investigated technologies of the system from metrological point of view, aiming to extract different error sources of the system. It also included short review of methods for error analysis and quality assessment.

The results show that the data quality of the sensor is much affected by the processing solution developed by Apple. The system combines flash LiDAR depth data with colour images, and the process seems to apply substantial smoothing to the depth data. The smoothing process tends to remove small details from the data, and thus the system has problems to capture those features in the data. On the other hand, applications of the system are strongly limited by the effective range of the device. As the upper limit is five meters, the system can perform best in indoor environments.

As said in the introduction, the depth sensor has been added to the examined device mainly to enhance AR experience and support imaging capabilities. These applications do not suffer from the found limitations, especially from the depth image smoothing, in similar way as shape acquisition applications. Instead, the continuity of the data flow and capability to segment significant objects from the scene is emphasized. Thus, the development of the system most likely does not focus on improving the system from this part. However, the situation could improve significantly with small additions to the API, such as access to the raw depth data for developers. Nevertheless, the measurement capabilities may improve in future also due to the development of the sensor technology, which is intensive due to wide application field and popularity of those applications (Dummer et al., 2021).

The experiments did not show any notable relationship between target material and measurement precision. However, the experimental setup did not really push the system to its limits, and different results may come out in different conditions, and different results might have appeared if the device was challenged more, e.g., by increasing the distance to the target. The relationship between the reflectance of the target material and accuracy is strongly expected, as the material properties define how much photons scatter back to the sensor, which has direct impact to the ability to distinguish the range information.

The most important note of the work is that the data captured within the effective range of the system seems to be relatively accurate with standard deviation less than three millimetres. However, the system is not capable to capture details of target scene. The capability is improved with assistance of RGB-camera feed, which seems to help segmentation of small objects from the background, though the quality of produced data is compromised. In measurement purposes, the system seems to suffer from intensive smoothing of depth maps, which the system performs to support AR experiences the system is built for.

Finally, it seems that the system has potential for simple measurement applications, but it cannot provide survey grade data. Examples of such applications are preparation of a floor plan of an existing apartment, data capture to support 3D modelling, or quick documentation of existing pipe network in industrial construction site to help to plan renovations. Major advantages of the system in these applications are affordability, ease of use, and time efficiency. In these applications, artifacts in the data are not that critical, as the product is constructed through human interpretation.

In further work, it would be interesting to see how distortions of the depth sensing behave on other units of the same device. A study optimizing filtering methods and data cleaning during capture is necessary. Also, there is a need for an assessment of the positioning algorithm, as it presumably has major effect to the quality of a continuously collected point cloud. A study similar to this work could also be performed with some Android device comprising ToF-sensor, as the ARCore API seems to allow better access for developers to the raw depth information (Google LLC, 2021). However, if such analysis is performed, it is important to note that the results are not directly comparable to this work as the investigated data is different due to the LiDAR and image aggregation the ARKit API performs.

References

Alexiou E & Ebrahimi T. (2018). *Point Cloud Quality Assessment Metric Based on Angular Similarity*. In: 2018 IEEE International Conference on Multimedia and Expo. San Diego, CA, USA. 23.-27.7.2018. IEEE. ISBN 1538617374. Available: DOI 10.1109/ICME.2018.8486512.

Apple Inc. (2021). *Apple Developer Documentation*. Available: https://developer.apple.com/documentation/ (Accessed 7.5.2021).

Apple Inc. (2020a). *Apple unveils new iPad Pro with breakthrough LiDAR Scanner and brings trackpad support to iPadOS* [Press release]. 18th March. Available: https://www.apple.com/newsroom/2020/03/apple-unveils-new-ipad-pro-with-lidar-scanner-and-trackpad-support-in-ipados/ (Accessed 7.5.2021).

Apple Inc. (2020b). *Explore ARKit 4*. In: 2020 Apple Worldwide Developers Conference. Held online, 22.-26.6.2020. Available: https://developer.apple.com/videos/play/wwdc2020/10611/ (Accessed 2.6.2021).

Beheim G & Klaus Fritsch. (1986). *Range finding using frequency-modulated laser diode*. Applied Optics, Vol. 25(9):1439. Available: DOI 10.1364/AO.25.001439.

Berendsen H. (2011). *A student's guide to data and error analysis*. New York, United States: Cambridge University Press. 225 p. ISBN 9781139079822.

Bradski G. (2000). *The openCV library*. Dr. Dobb's Journal: Software Tools for the Professional Programmer, Vol. 25(11):120.

Chen G, Wiede C & Kokozinski R. (2021). *Data Processing Approaches on SPAD-Based d-TOF LiDAR Systems: A Review*. IEEE Sensors Journal. Vol. 21(5):5656. Available: DOI 10.1109/JSEN.2020.3038487.

Dummer M, Johnson K, Rothwell S, Tatah K, & Hibbs-Brenner M. (2021). The role of VCSELs in 3D sensing and LiDAR. In: Chen R & Schröder H. Optical Interconnects XXI. 5.3.2021. SPIE, 2021. Vol. 116920. ISBN 9781510642195. Available: DOI 10.1117/12.2577885.

Giancola S, Valenti M & Sala R. (2018). A survey on 3D cameras: Metrological comparison of time-of-flight, structured-light and active stereoscopy technologies. Cham: Springer Nature. 90 p. ISBN 9783319917610.

Gollob C, Ritter T, Kraßnitzer R, Tockner A & Nothdurft A. (2021). *Measurement of Forest Inventory Parameters with Apple iPad Pro and Integrated LiDAR Technology*. Remote Sensing, Vol. 13(16):3129. Available: DOI 10.3390/rs13163129.

Google LLC. (2021). *Use Raw Depth in your Android app,* in ARCore documentation. Available: https://developers.google.com/ar/develop/java/depth/raw-depth. (Accessed 18.10.2021).

Grzegorzek M, Theobalt C, Koch R & Kolb A. (2013). *Time-of-Flight and Depth Imaging. Sensors, Algorithms, and Applications*. Springer. 319 p. ISBN 9783642449642.

Han S, Meng Z, Omisore O, Akinyemi T, Yan Y. (2020). *Random Error Reduction Algorithms for MEMS Inertial Sensor Accuracy Improvement—A Review*. Micromachines, Vol. 11(11):1021. Available: DOI 10.3390/mi11111021.

Harris C, Millman K, van der Walt S, Gommers R, Virtanen P, Cournapeau D, Wieser E, Taylor J, Berg S, Smith N J, Kern R, Picus M, Hoyer S, van Kerkwijk M H, Brett M, Haldane A, Fernández del Río J, Wiebe M, Peterson P, Gérard-Marchant P, Sheppard K, Reddy T, Weckesser W, Abbasi H, Gohlke C & Oliphant T E. (2020). *Array programming with NumPy*. Nature, Vol. 585:357. Available: DOI 10.1038/s41586-020-2649-2.

Heinrichs B & Yang M. (2021). *Bias and Repeatability of Measurements from 3D Scans Made Using iOS-Based Lidar*. SAE Technical Paper 2021-01-0891. Available: DOI 10.4271/2021-01-0891.

Ingman M, Virtanen J P, Vaaja M T, & Hyyppä, H. (2020). *A comparison of low-cost sensor systems in automatic cloud-based indoor 3D modeling*. Remote Sensing, Vol. 12(16):2624. Available: DOI 10.3390/RS12162624.

JCGM 100:2008. (2008). Evaluation of measurement data — Guide to the expression of uncertainty in measurement. Corrected version 2010. 120 p. Available online: https://www.bipm.org/documents/20126/2071204/JCGM_100_2008_E.pdf (Accessed 10.11.2021).

Lange R, & Seitz P. (2001). *Solid-state time-of-flight range camera*. IEEE Journal of Quantum Electronics, Vol. 37(3):390. Available: DOI 10.1109/3.910448.

Lenovo Ltd. (2016). *Lenovo Unveils World's First Tango-Enabled Smartphone – PHAB2 Pro* [Press Release]. 19th June. Available: https://news.lenovo.com/pressroom/press-releases/lenovo-unveils-worlds-first-tango-enabled-smartphone-phab2-pro/ (Accessed 2.11.2021).

Leutenegger S, Lynen S, Bosse M, Siegwart R & Furgale P. (2015). *Keyframe-based visual–inertial odometry using nonlinear optimization*. The International Journal of Robotics Research, Vol. 34(3):314. Available: DOI 10.1177/0278364914554813.

Li M & Mourikis A. (2013). *High-precision, consistent EKF-based visual-inertial Odometry*. International Journal of Robotics Research, Vol. 32(6):690. Available: DOI 10.1177/0278364913481251.

Luetzenburg G, Kroon A. & Bjørk A A. (2021). *Evaluation of the Apple iPhone 12 Pro LiDAR for an Application in Geosciences*. Nature Scientific Reports, Vol. 11. Available: DOI 10.1038/s41598-021-01763-9.

Murtiyoso A, Grussenmeyer P, Landes T & Macher H. (2021). *First Assessments Into the Use of Commercial-Grade Solid State LIDAR for Low Cost Heritage Documentation*. ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. 43:599. Available: DOI 10.5194/isprs-archives-XLIII-B2-2021-599-2021.

Nistér D, Naroditsky O & Bergen J. (2004). *Visual odometry*. In: Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. Wshington DC, USA. 27.6.-2.7.2004. IEEE. Vol. 1. pp. 646-652. ISBN: 0769521584. Available: DOI 10.1109/CVPR.2004.1315094.

Poddar S, Kumar V & Kumar A. (2016). *A Comprehensive Overview of Inertial Sensor Calibration Techniques*. Journal of Dynamic Systems Measurement and Control, Vol. 139(1):11006. Available: DOI 10.1115/1.4034419.

Remondino F & Stoppa D. (2013). *TOF range-imaging cameras*. Springer. 240 p. ISBN 9783642275227.

Samsung Electronics Co. Ltd. (2020). *Introducing the Samsung Galaxy S20: Change the Way You Experience the World* [Press Release]. 12th February. Available: https://news.samsung.com/global/introducing-the-samsung-galaxy-s20-change-the-way-you-experience-the-world (Accessed 2.11.2021).

Schubert D, Goll T, Demmel N, Usenko V, Stückler J & Cremers D. (2018). *The TUM VI Benchmark for Evaluating Visual-Inertial Odometry*. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Madrid, Spain. 1.-5.10.2018. pp. 1680-1687. IEEE. ISBN 9781538680957. Available: DOI 10.1109/IROS.2018.8593419.

Shan J, Toth C. (2018). *Topographic Laser Ranging and Scanning: Principles and Processing*. 2nd ed. Milton: Taylor & Francis Group. 638 p. ISBN: 9781498772273.

Stray Scanner (Version 1.2) [Software]. (2020). Available: https://apps.apple.com/fi/app/stray-scanner/id1557051662.

US 20190362511A1. (2019). *Efficient scene depth map enhancement for low power devices*. Apple Inc., Cupertino, California, United States. (Norman M, Tao M, Bujold E, Soussan S, Roelke V, Anneheim G, Zaragoza J & Ciurea F). 15987834, 23.5.2018. 28.11.2019. 27 p.

US 2020158837A1. (2020). *SPAD array with gated histogram construction*. Apple Inc., Cupertino, California, United States. (Sharma A, Laflaquière A, Agranov G, Rosenblum G & Mandai S) 16752653, 26.1.2020. 21.5.2020. 12 p.

US 2020256993A1. (2020). *Depth sensing using a sparse array of pulsed beams*. Apple Inc., Cupertino, California, United States. (Oggier T) 16532513, 6.8.2019. 13.8.2020. 17 p.

Vermeer M. (2019). *Geodesy: The science underneath*. Aalto University publication series SCIENCE + TECHNOLOGY. 605 p. ISBN: 9789526088723. Available: http://urn.fi/URN:ISBN:978-952-60-8872-3.

Virtanen P, Gommers R, Oliphant T E, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt S J, Brett M, Wilson J, Millman K J, Mayorov N, Nelson A R J, Jones E, Kern R, Larson E, Carey C J, Polat İ, Feng Y, Moore E W, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero E A, Harris C R, Archibald A M, Ribeiro A H, Pedregosa F, van Mulbregt P & SciPy 1.0 Contributors. (2020). *SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python*. Nature Methods, Vol. 17(3):261. Available: DOI 10.1038/s41592-019-0686-2.

Vogt M, Rips A & Emmelmann C. (2021). *Comparison of iPad Pro®'s LiDAR and TrueDepth Capabilities with an Industrial 3D Scanning Solution*. Technologies. Vol. 9(2):25. Available: DOI 10.3390/technologies9020025.

Vosselman G & Maas H-G. (2010). *Airborne and Terrestrial Laser Scanning*. Dunbeath: Whittles Publishing. 337 p. ISBN 9781849950138.

Weinmann M. (2016). Reconstruction and Analysis of 3D Scenes: From Irregularly Distributed 3D Points to Object Classes. Springer International Publishing. 233 p. ISBN 9783319292441.

Westin S H, Li H & Torrance K E. (2004). *A comparison of four brdf models*. In Eurographics Symposium on Rendering.

Wang Z, Bovik A C, Sheikh H R & Simoncelli E P. (2004). *Image quality assessment: from error visibility to structural similarity*. IEEE Transactions on Image Processing, Vol. 13(4):600. Available: DOI 10.1109/TIP.2003.819861.

Zhou Q Y, Park J & Koltun V. (2018). *Open3D: A modern library for 3D data processing*. arXiv preprint. Available: arXiv:1801.09847.

APPENDIX A: Data processing scripts

Part of Master's Thesis in Aalto University

The aim of these expriments is to find out the data collection accuracy of the examined sensor. First, some experiments are done to find out best methods to perform final testing. Then, the actual benchmarking follows. The aim of these tests is to find out capabilities of the flash LiDAR sensor, so the intention is to leave RGB-camera data unused.

Numbering of the parts of this appendix corresponds to the numbering of subsection ***Results and discussion*** in the work.

First, import tools and functionalities and configure settings. Helper functions are attached to the end of this document.

In [ ]:
```
import os
import numpy as np
import matplotlib.pyplot as plt
import open3d as o3d
import cv2 as cv

from mpl_toolkits.axes_grid1 import make_axes_locatable
from scipy.interpolate import griddata
import scipy.stats

%matplotlib inline
  #instead of widget for export purposes
plt.rcParams['figure.figsize'] = (12, 8)

from helpers.read_data import *
from helpers.process_data import *

data_folder = '../Data'
```

4.1 Illumination pattern and intensity distribution

The aim of following snapshots is to point out how the illumination power of the sensor is distributed.

The process begins by loading data

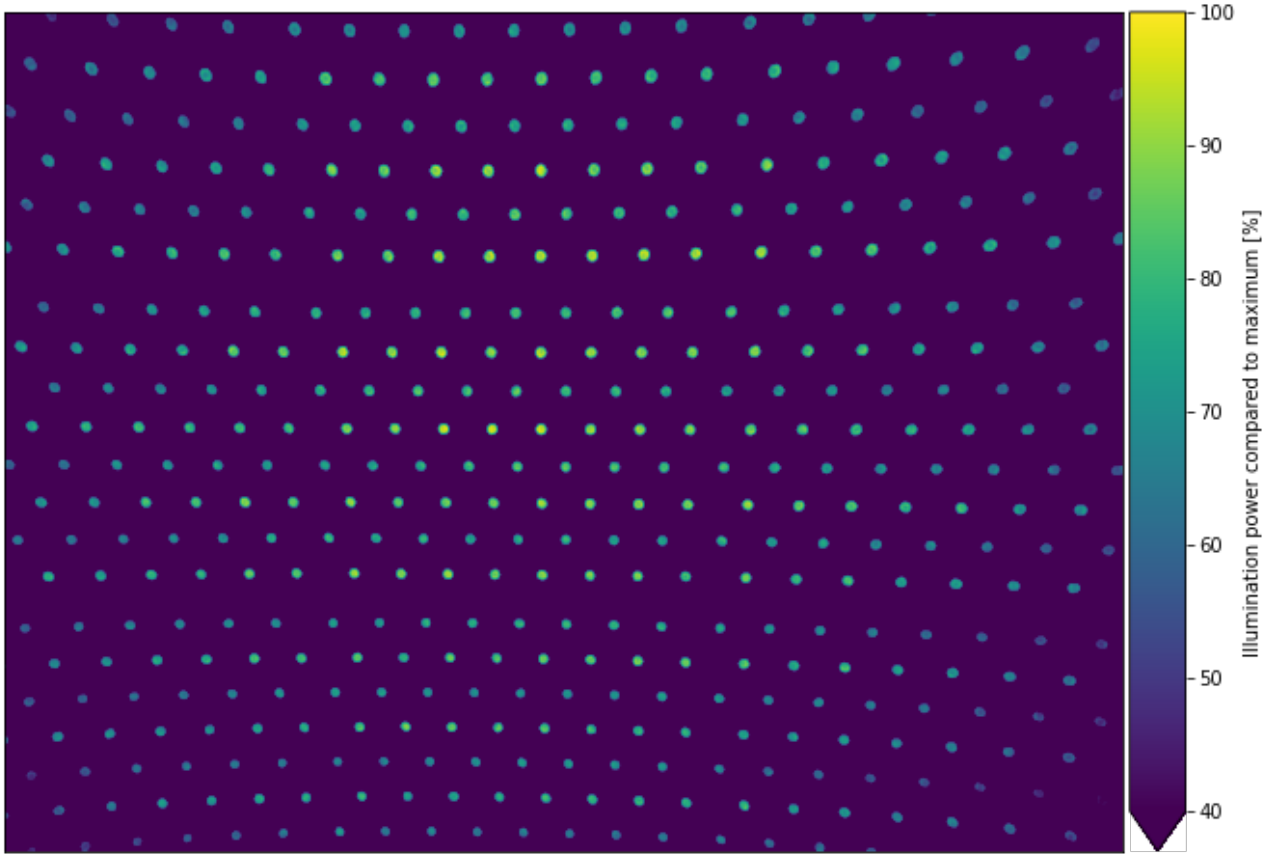
In [ ]:
```
intensity_data = read_intensity_data(data_folder + "/intensity/Direct")
validation_data = read_intensity_data(data_folder + "/intensity/Tilted")
intensity_data.shape
```

Out[ ]:
```
(2448, 3264)
```

Visualizing data

In [ ]:

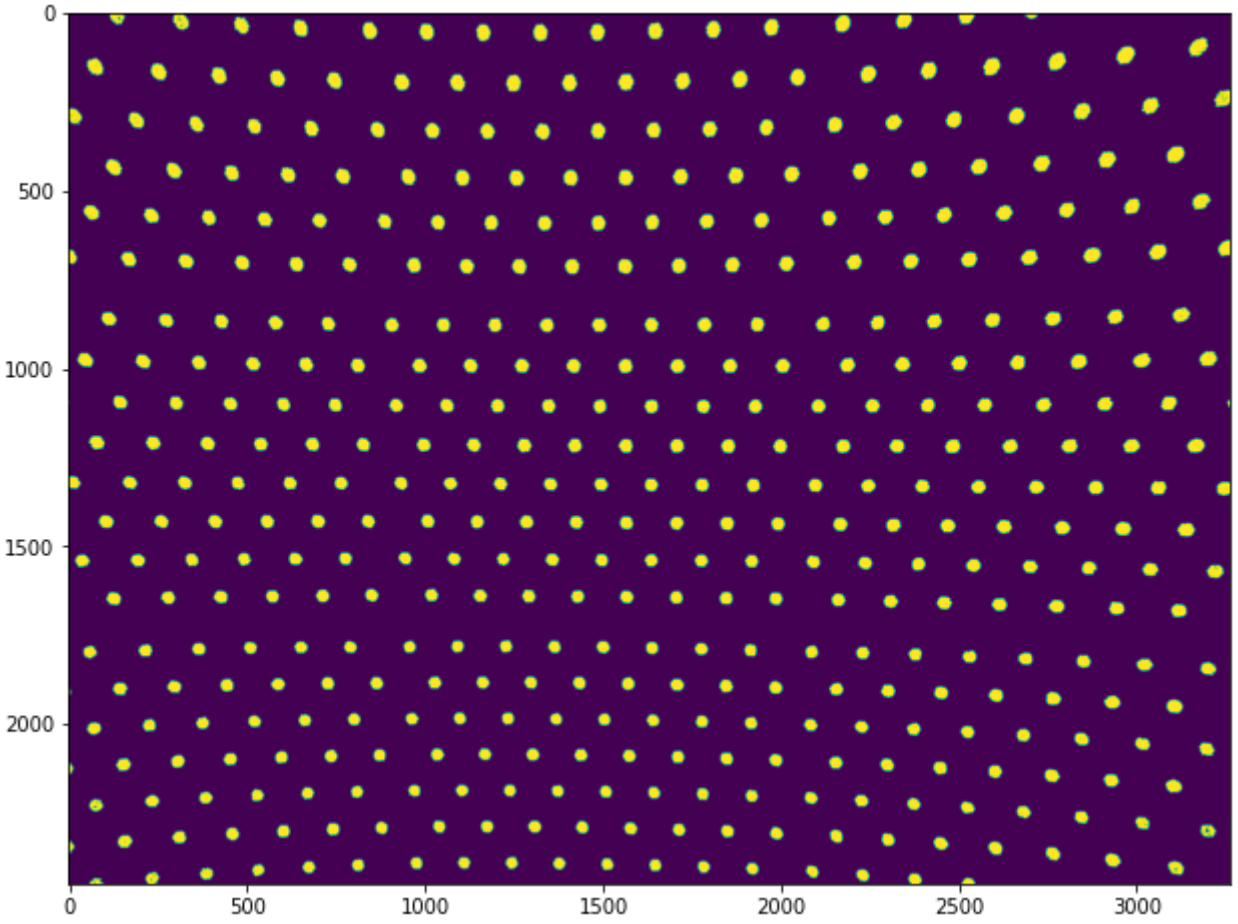
```
plt.close()
smooth_intensity = cv.GaussianBlur(intensity_data, (25, 25), 0)
im = plt.imshow(smooth_intensity / smooth_intensity.max() * 100, vmin=40)

plt.tick_params(
    axis="both",
    which="both",
    bottom=False,
    labelbottom=False,
    left=False,
    labelleft=False,
)

divider = make_axes_locatable(plt.gca())
cax = divider.append_axes("right", size="5%", pad=0.05)
cbar = plt.colorbar(im, cax=cax, extend="min")
cbar.set_label("Illumination power compared to maximum [%]")
```



Thresholding image for segmentation, detecting individual dots, and interpolating intensity field

In [ ]:

```
plt.close()
illuminated = intensity_data > 50
# Thresholding intensity image,
# expecting all dots have higher intensity than 50
kernel = np.ones((3, 3), np.uint8)
illuminated = cv.morphologyEx(
    illuminated.astype("uint8"), cv.MORPH_OPEN, kernel, iterations=2
)

plt.imshow(illuminated)
# Visualizing thresholding result
```



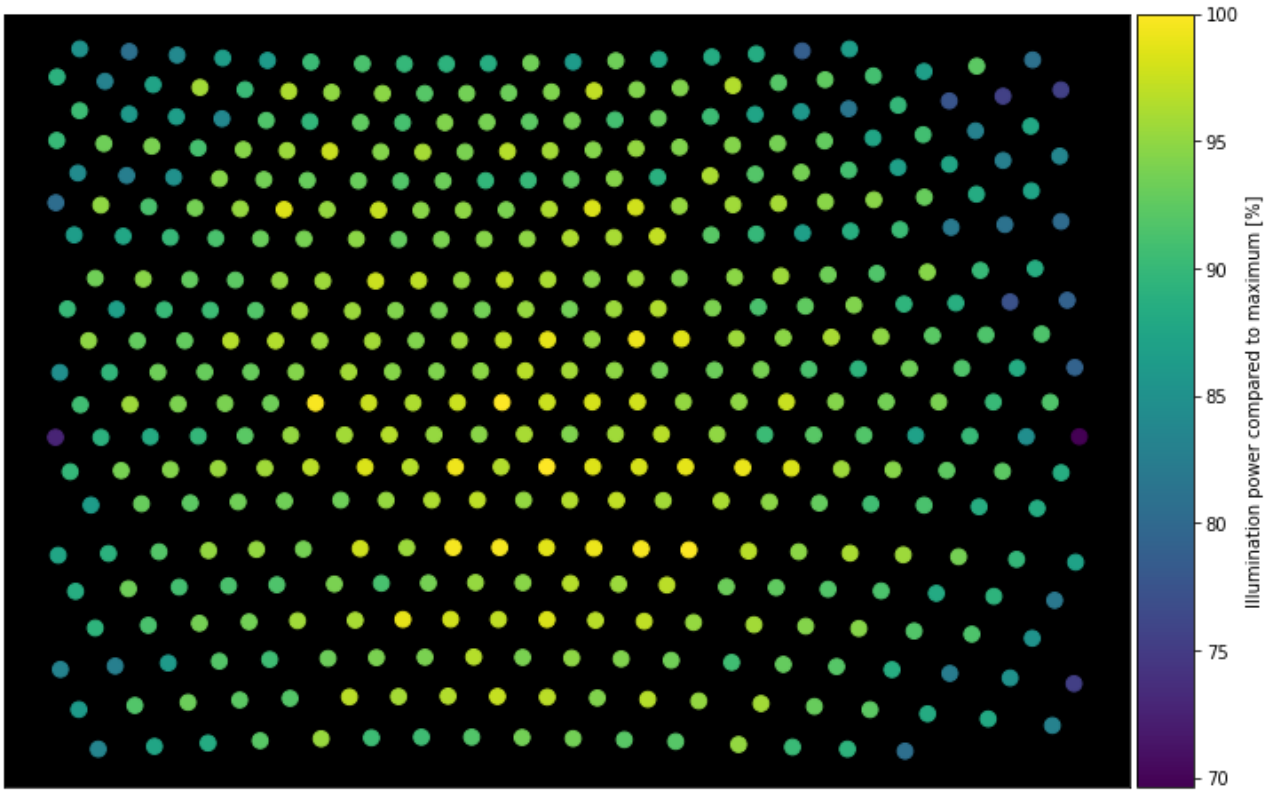
In [ ]:

```
n_labels, labels, _, centroids = cv.connectedComponentsWithStats(illuminated)
# Detecting individual illumination points

intensities = []
for i in range(1, n_labels):  # Finding maximum intensity for each label
    intensities.append(intensity_data[labels == i].max())
intensities_scaled = np.asarray(intensities) / max(intensities) * 100
# Scaling intensities to range 0-100

grid_x, grid_y = np.mgrid[0:3264, 0:2448]
interpolatedIntensityField = griddata(
    centroids[1:], intensities_scaled, (grid_x, grid_y), method="linear"
)  # Interpolating continuous field for visualization
```

Visualizing individual laser dots

In [ ]:

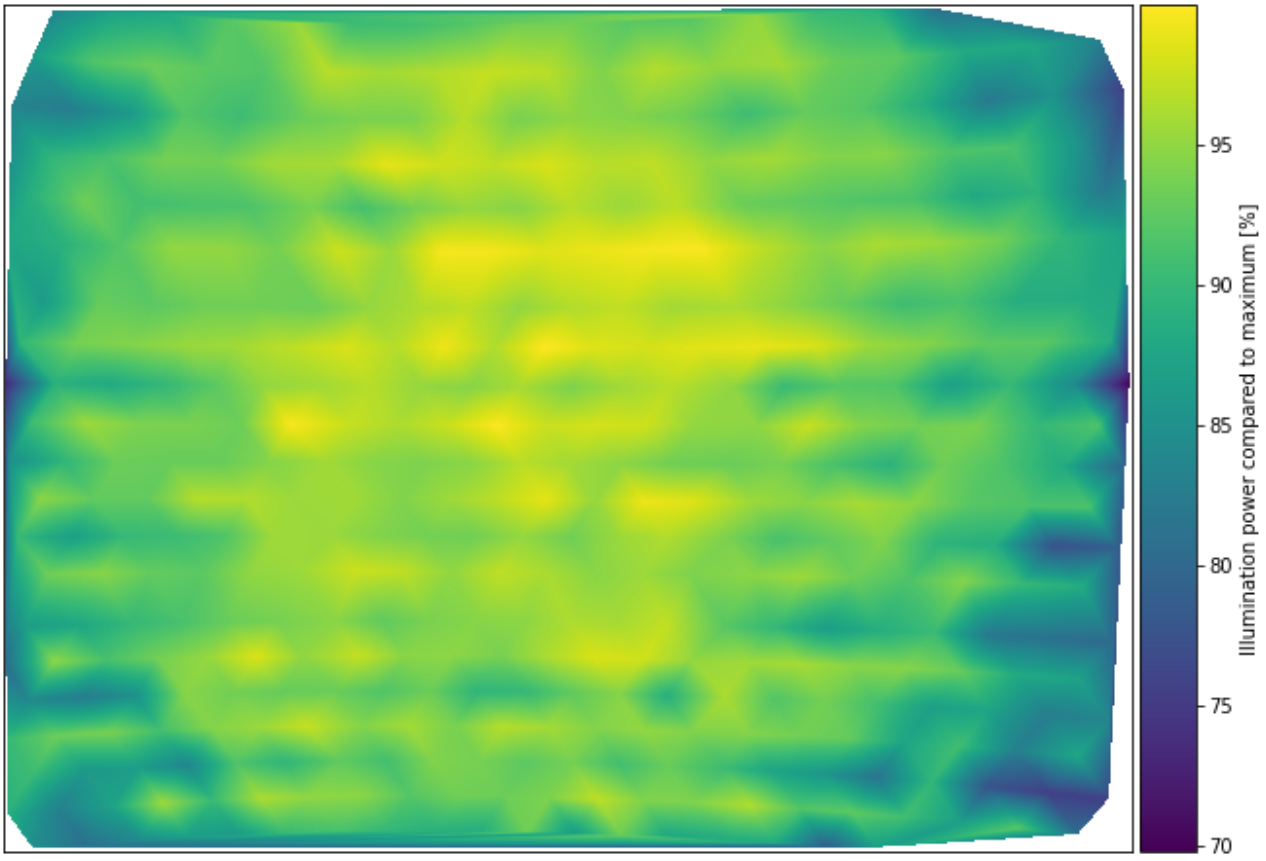
```
plt.close()
ax = plt.subplot(facecolor="black")
sc = plt.scatter(
    x=centroids[1:, 0], y=centroids[1:, 1], c=intensities_scaled, s=75
)
plt.tick_params(
    axis="both",
    which="both",
    bottom=False,
    labelbottom=False,
    left=False,
    labelleft=False,
)

divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cbar = plt.colorbar(sc, cax=cax)
cbar.set_label("Illumination power compared to maximum [%]")
```



Visualizing interpolated intensity field

In [ ]:

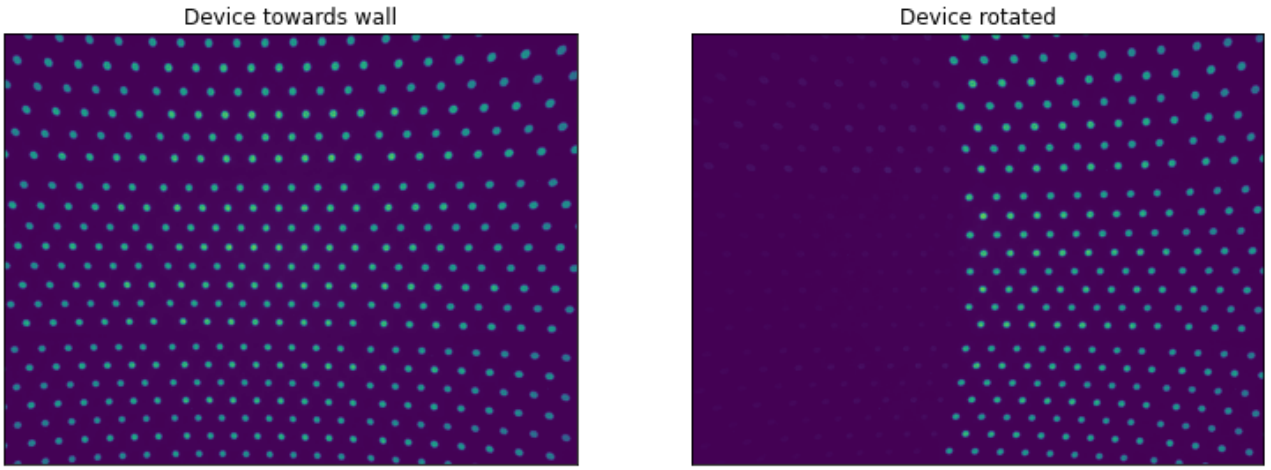
```
plt.close()
i_fig = plt.figure()
i_ax = plt.subplot()
i_im = i_ax.imshow(
    interpolatedIntensityField.T
)
plt.tick_params(
    axis="both",
    which="both",
    bottom=False,
    labelbottom=False,
    left=False,
    labelleft=False,
)

divider = make_axes_locatable(i_ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cbar = plt.colorbar(i_im, cax=cax)
cbar.set_label("Illumination power compared to maximum [%]")
```



Performing the the same workflow for images from tilted imaging position. The intention is to observe effect of the light incidence angle.

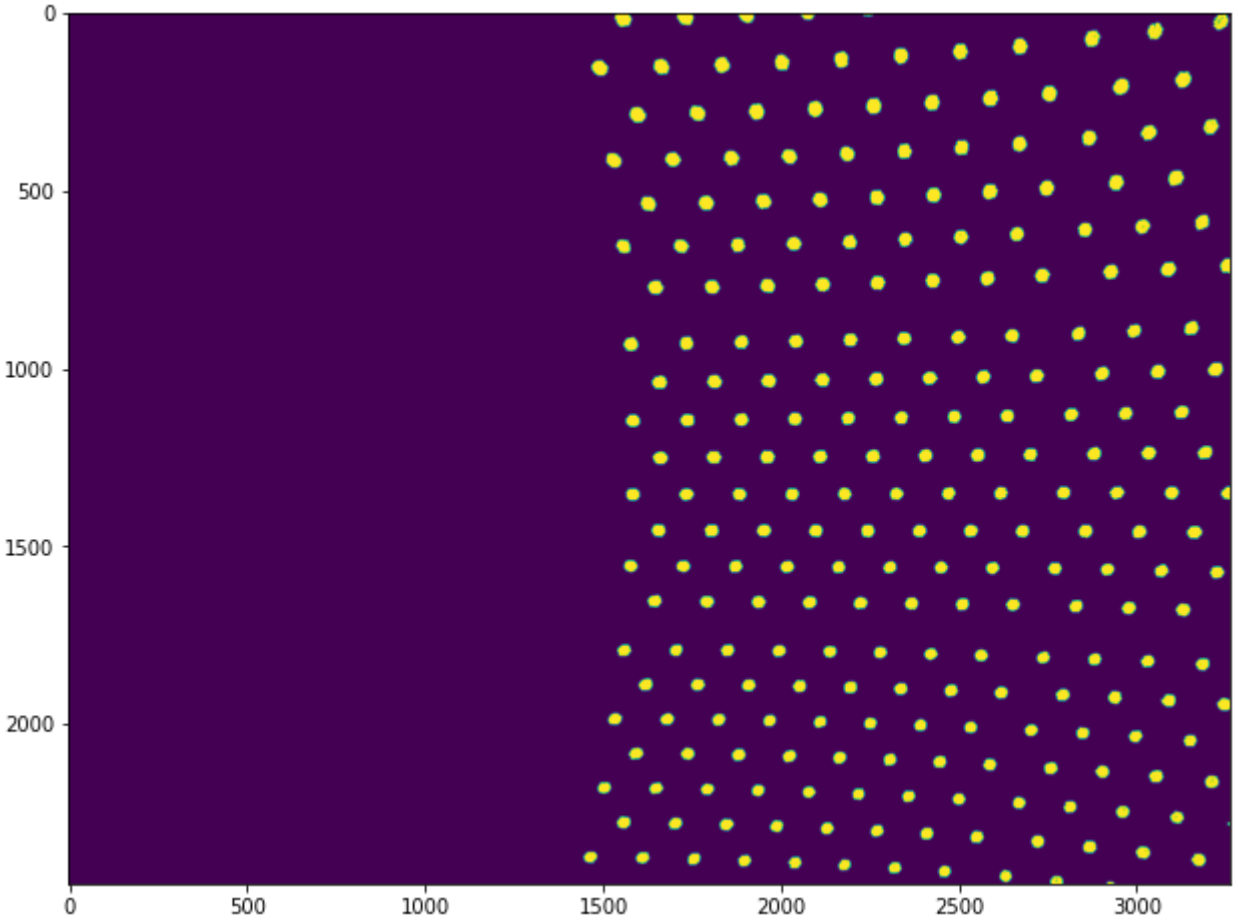
In [ ]:

```
plt.close()
plt.figure(figsize=(12.8, 4.8))
direct = plt.subplot(121)
direct.set_title("Device towards wall")
im = plt.imshow(intensity_data)
plt.tick_params(
    axis="both",
    which="both",
    bottom=False,
    labelbottom=False,
    left=False,
    labelleft=False,
)

tilted = plt.subplot(122)
tilted.set_title("Device rotated")
im2 = plt.imshow(validation_data)
im2.set_clim(im.get_clim())
plt.tick_params(
    axis="both",
    which="both",
    bottom=False,
    labelbottom=False,
    left=False,
    labelleft=False,
)
```



In [ ]:

```
illuminated_validation = validation_data > 50
# Thresholding intensity image, expecting all dots have higher intensity than 50
illuminated_validation = cv.morphologyEx(
    illuminated_validation.astype("uint8"), cv.MORPH_OPEN, kernel, iterations=2
)

plt.imshow(illuminated_validation);
# Visualizing thresholding result
```

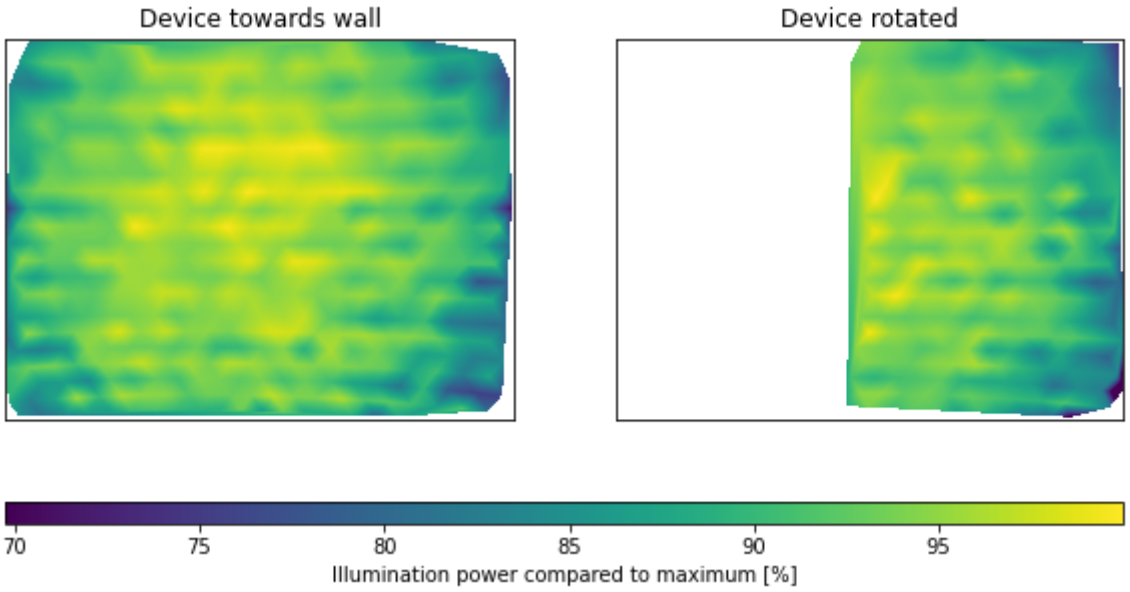


In [ ]:

```
val_n_labels, val_labels, _, val_centroids = cv.connectedComponentsWithStats(
    illuminated_validation
)  # Detecting individual illumination points

intensities_val = []
for i in range(1, val_n_labels):  # Finding maximum intensity for each label
    intensities_val.append(validation_data[val_labels == i].max())
val_intensities_scaled = np.asarray(intensities_val) / max(intensities) * 100
# Scaling intensities to be comparable with earlier data

val_interpolatedIntensityField = griddata(
    val_centroids[1:], val_intensities_scaled, (grid_x, grid_y), method="linear"
)  # Interpolating continuous field for visualization
```

In [ ]:

```
plt.close()
plt.figure(figsize=(10, 5))
direct = plt.subplot(121)
direct.set_title("Device towards wall")
im = plt.imshow(interpolatedIntensityField.T)
plt.tick_params(
    axis="both",
    which="both",
    bottom=False,
    labelbottom=False,
    left=False,
    labelleft=False,
)

tilted = plt.subplot(122)
tilted.set_title("Device rotated")
im2 = plt.imshow(val_interpolatedIntensityField.T)
im2.set_clim(im.get_clim())
plt.tick_params(
    axis="both",
    which="both",
    bottom=False,
    labelbottom=False,
    left=False,
    labelleft=False,
)

cbar = plt.gcf().colorbar(
    im, ax=[direct, tilted], orientation="horizontal", aspect=50
)
cbar.set_label("Illumination power compared to maximum [%]")
```



4.2 Importance and effect of camera feed in depth map creation

Following snapshots try to point out how much camera feed affects depth map creation.

In this experiment, a simple scene was built containing a printed picture of geometry, simple block geometry (block of post-it badges) and a bit more detailed objects (a couple of pens in a cup). The sensor was pointed to this scene and data collection was started. After a couple of seconds of data

In [ ]:
```
depth_data, confidence_data, rgb_data = read_data(
    folder=data_folder + "/0_2", depth_maps=True, rgb_frames=True
)
imageIR = np.asarray(Image.open(data_folder + "/0_2/IMG_0630.JPG"))

rgb_data.shape
```

Out[ ]:
```
(948, 1440, 1920, 3)
```

Visualizing comparison of rgb video and images containing infrared light

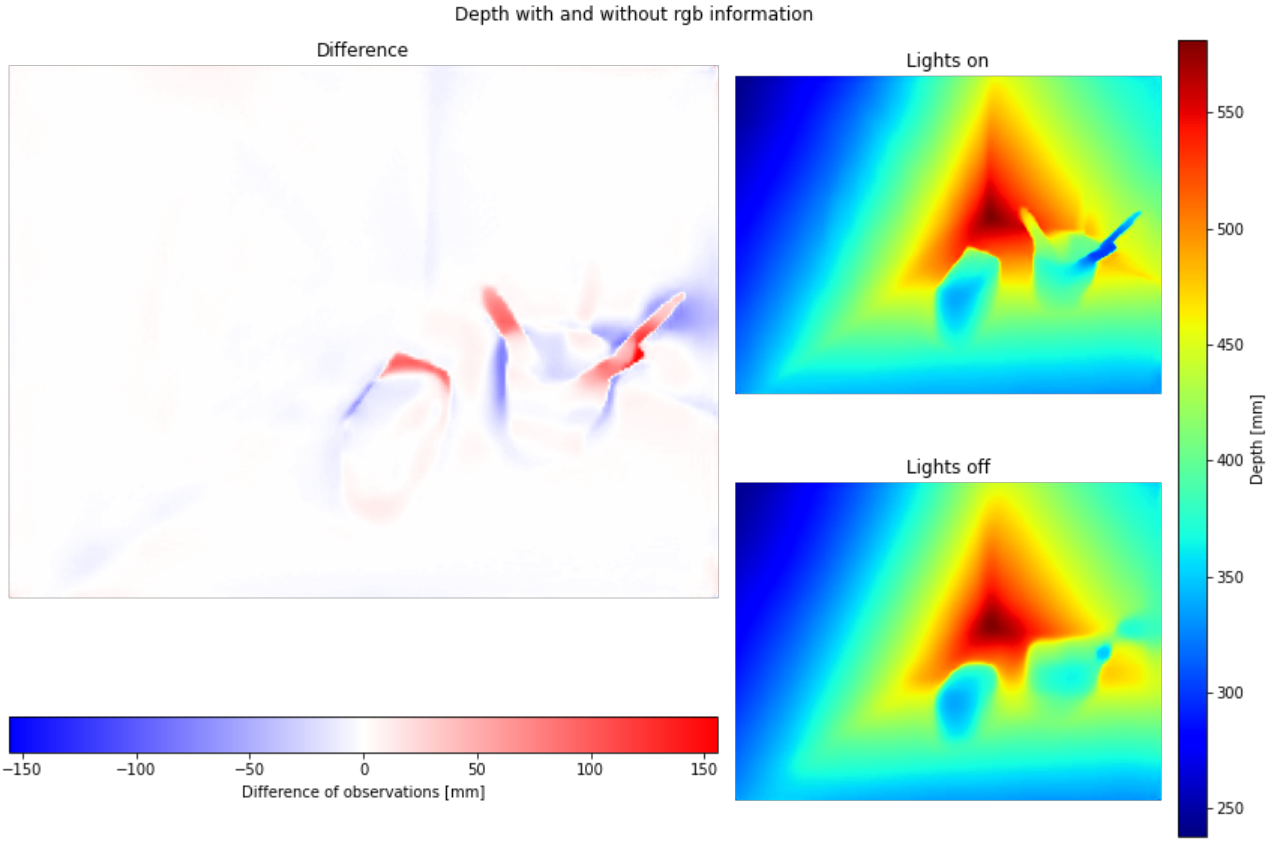
In [ ]:
```
plt.close()

ax = plt.subplot(1, 2, 1)
plt.axis("off")
plt.imshow(rgb_data[-60])
ax.set_title("iPad rgb feed")

ax = plt.subplot(1, 2, 2)
plt.axis("off")
plt.imshow(imageIR)
ax.set_title("IR camera frame")
```

Extracting two separate cases from the continuous data and computing median depths and deviations.

In [ ]:
```
time_offset = 1
duration = 5
depths_lightsOn = depth_data[
    -((time_offset + duration) * 60) : -(time_offset * 60)
]
depths_lightsOff = depth_data[
    (time_offset * 60) : ((time_offset + duration) * 60)
]

depth_median_lightsOn = np.median(depths_lightsOn, axis=0)
depth_median_lightsOff = np.median(depths_lightsOff, axis=0)

depth_std_lightsOn = np.std(depths_lightsOn, axis=0)
depth_std_lightsOff = np.std(depths_lightsOff, axis=0)

depth_std_diff = depth_std_lightsOff - depth_std_lightsOn
depth_median_diff = depth_median_lightsOff - depth_median_lightsOn
```

In [ ]:

```
vmax = np.max([depth_median_lightsOn.max(), depth_median_lightsOff.max()])
vmin = np.min([depth_median_lightsOn.min(), depth_median_lightsOff.min()])

plt.close()

fig = plt.figure(tight_layout=True)
gs = fig.add_gridspec(2, 3, width_ratios=[2.5, 1.5, 0.1])
axs = []

axs.append(fig.add_subplot(gs[0, 1]))
axs[-1].imshow(depth_median_lightsOn, cmap="jet", vmax=vmax, vmin=vmin)
axs[-1].set_title("Lights on")

axs.append(fig.add_subplot(gs[1, 1]))
im = axs[-1].imshow(depth_median_lightsOff, cmap="jet", vmax=vmax, vmin=vmin)
axs[-1].set_title("Lights off")

axs.append(fig.add_subplot(gs[:, 0]))
diff_im = axs[-1].imshow(
    depth_median_diff,
    cmap="bwr",
    vmax=np.abs(depth_median_diff).max(),
    vmin=-np.abs(depth_median_diff).max(),
)
axs[-1].set_title("Difference")

for axis in axs:
    axis.set_axis_off()

plt.suptitle("Depth with and without rgb information")
axs.append(fig.add_subplot(gs[:, -1]))
fig.colorbar(im, cax=axs[-1], label="Depth [mm]")
fig.colorbar(
    diff_im,
    ax=axs[2],
    orientation="horizontal",
    label="Difference of observations [mm]",
);
```



Displaying the data interactively in point cloud format

In [ ]:
```
intrinsics = read_intrinsics(file=data_folder + "/0_2/camera_matrix.csv")
pointcloud = o3d.geometry.PointCloud.create_from_depth_image(
    o3d.geometry.Image(depth_median_lightsOn.astype("uint16")), intrinsics
)
o3d.visualization.draw_geometries([pointcloud])
```

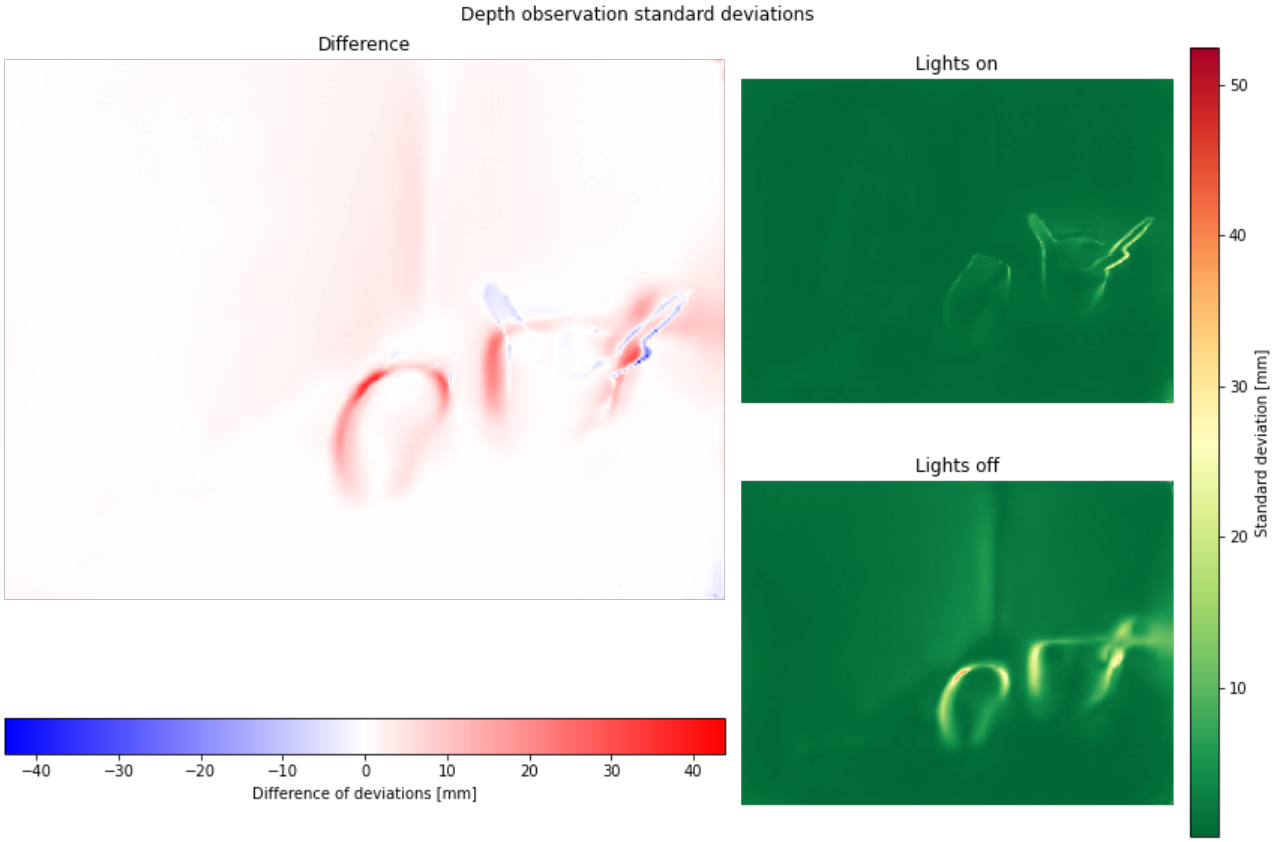
A cell for preparing prints of the point cloud from a specified point of view

In [ ]:
```
vis = o3d.visualization.Visualizer()
vis.create_window()
vis.add_geometry(pointcloud)
parameters = o3d.io.read_pinhole_camera_parameters(
    "ScreenCamera_2021-09-29-08-57-52.json"
)
ctr = vis.get_view_control()
ctr.convert_from_pinhole_camera_parameters(parameters)
vis.run()
vis.destroy_window()
del ctr
```

Visualizing the deviations within both datasets and difference of those

In [ ]:

```
vmax = np.max([depth_std_lightsOn.max(), depth_std_lightsOff.max()])
vmin = np.min([depth_std_lightsOn.min(), depth_std_lightsOff.min()])

plt.close()

fig = plt.figure(tight_layout=True)
gs = fig.add_gridspec(2, 3, width_ratios=[2.5, 1.5, 0.1])
axs = []

axs.append(fig.add_subplot(gs[0, 1]))
axs[-1].imshow(depth_std_lightsOn, cmap="RdYlGn_r", vmax=vmax, vmin=vmin)
axs[-1].set_title("Lights on")

axs.append(fig.add_subplot(gs[1, 1]))
im = axs[-1].imshow(depth_std_lightsOff, cmap="RdYlGn_r", vmax=vmax, vmin=vmin)
axs[-1].set_title("Lights off")

axs.append(fig.add_subplot(gs[:, 0]))
diff_im = axs[-1].imshow(
    depth_std_diff,
    cmap="bwr",
    vmax=np.abs(depth_std_diff).max(),
    vmin=-np.abs(depth_std_diff).max(),
)
axs[-1].set_title("Difference")

for axis in axs:
    axis.set_axis_off()

plt.suptitle("Depth observation standard deviations")
axs.append(fig.add_subplot(gs[:, -1]))
fig.colorbar(im, cax=axs[-1], label="Standard deviation [mm]")
fig.colorbar(
    diff_im,
    ax=axs[2],
    orientation="horizontal",
    label="Difference of deviations [mm]",
);
```



4.3 Range measurement repeatability

This experiment pursues to find out the precision of range measurement as function of distance to the object. Some idea of the absolute accuracy may also come as side product. However, the distance of the device to the object is not absolutely accurate compared to the measurement accurracy of the device, so this information is slightly questionable.

Reading sample data to an array

In [ ]:

```
range_measurements = []

intervals = [0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5]

for folder in os.scandir(data_folder + "/half_meter_garage/"):
    if not folder.is_dir():
        continue  # Neglect files, read subfolders only
    print(folder.path)

    depth_data, _, _ = read_data(folder.path)
    range_measurements.append(
        depth_data[60:540, 192 // 2, 256 // 2]
    )  # Removing the effect of shaking by leaving points away from the ends,
       # using only central pixel values

range_measurements = np.asarray(range_measurements)
```

```
../Data/half_meter_garage/0_5m
../Data/half_meter_garage/1_0m
../Data/half_meter_garage/1_5m
../Data/half_meter_garage/2_0m
../Data/half_meter_garage/2_5m
../Data/half_meter_garage/3_0m
../Data/half_meter_garage/3_5m
../Data/half_meter_garage/4_0m
```

```
../Data/half_meter_garage/4_5m
../Data/half_meter_garage/5_0m
../Data/half_meter_garage/5_5m
../Data/half_meter_garage/6_0m
../Data/half_meter_garage/6_5m
../Data/half_meter_garage/7_0m
../Data/half_meter_garage/7_5m
```

Computing descriptive statistics of the sample data

In [ ]:

```
details = dict(
    median=np.median(range_measurements, axis=1),
    mean=np.mean(range_measurements, axis=1),
    std=np.std(range_measurements, axis=1),
    min=np.min(range_measurements, axis=1),
    max=np.max(range_measurements, axis=1),
    upper=np.percentile(range_measurements, 75, axis=1),
    lower=np.percentile(range_measurements, 25, axis=1),
    iqr=scipy.stats.iqr(range_measurements, axis=1),
)
```

Visualizing measurements

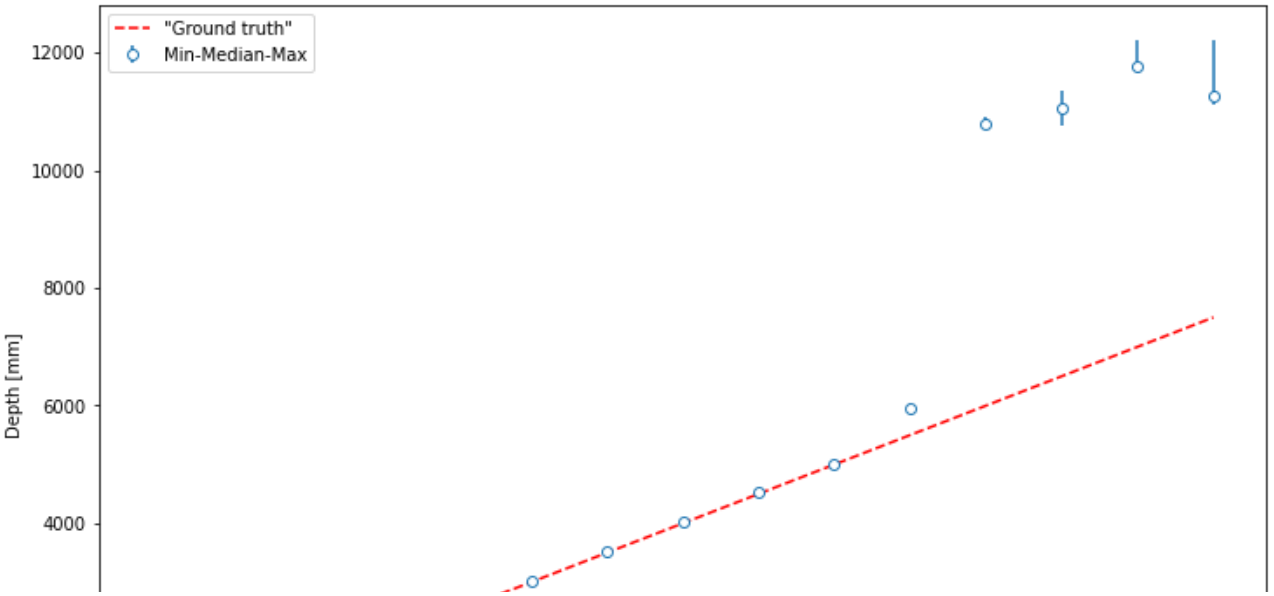
First, "real" distances versus measured distances

In [ ]:

```
plt.close()

minmax = [
    details["median"] - details["min"],
    details["max"] - details["median"],
]

plt.plot(
    intervals,
    [i * 1000 for i in intervals],
    color="r",
    ls="--",
    label='"Ground truth"',
)
plt.errorbar(
    intervals,
    details["median"],
    minmax,
    fmt="o",
    mfc="w",
    capthick=10,
    label="Min-Median-Max",
)

ax = plt.gca()
ax.set_xlabel("Distance from the object [m]")
ax.set_ylabel("Depth [mm]")
ax.legend()
```

Out[ ]:

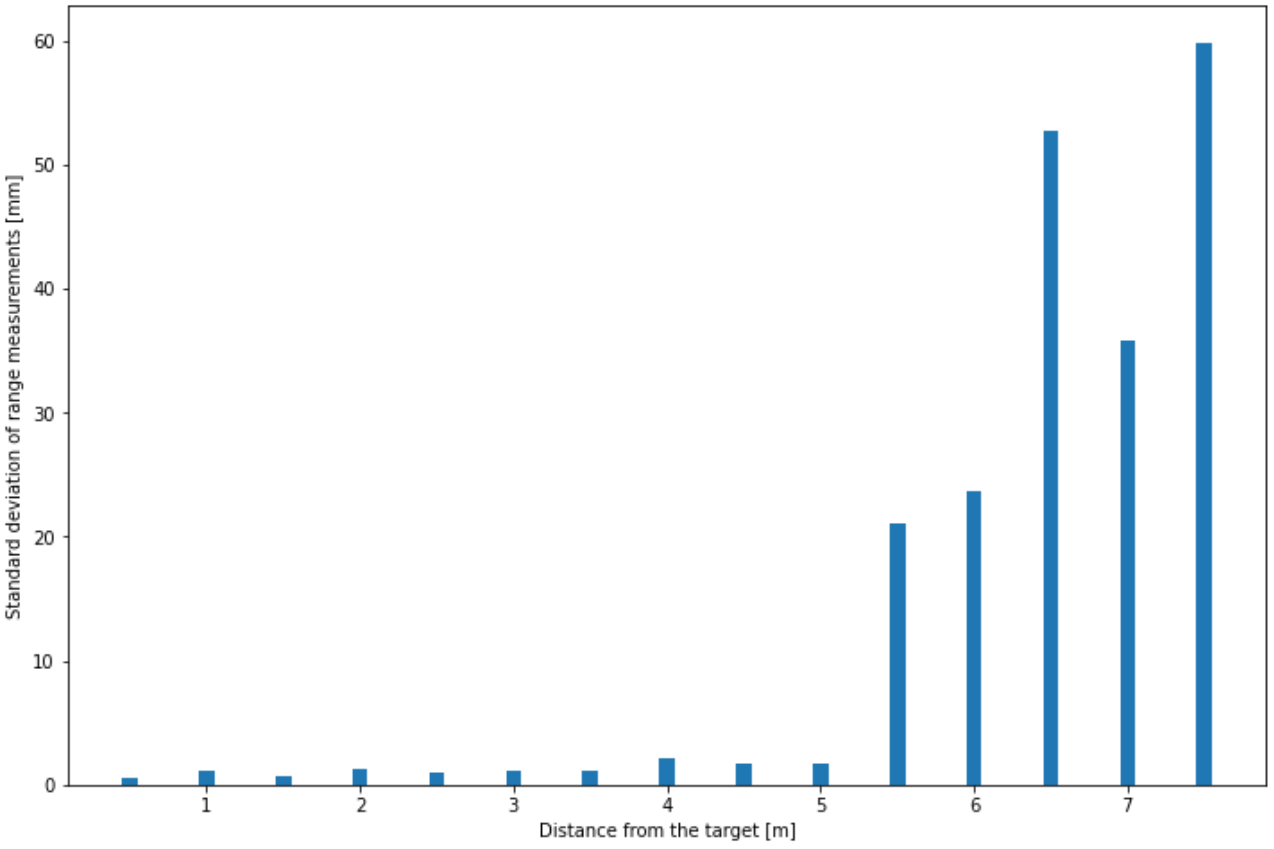
```
<matplotlib.legend.Legend at 0x18ec1cc1e80>
```



Standard deviations against distance

In [ ]:

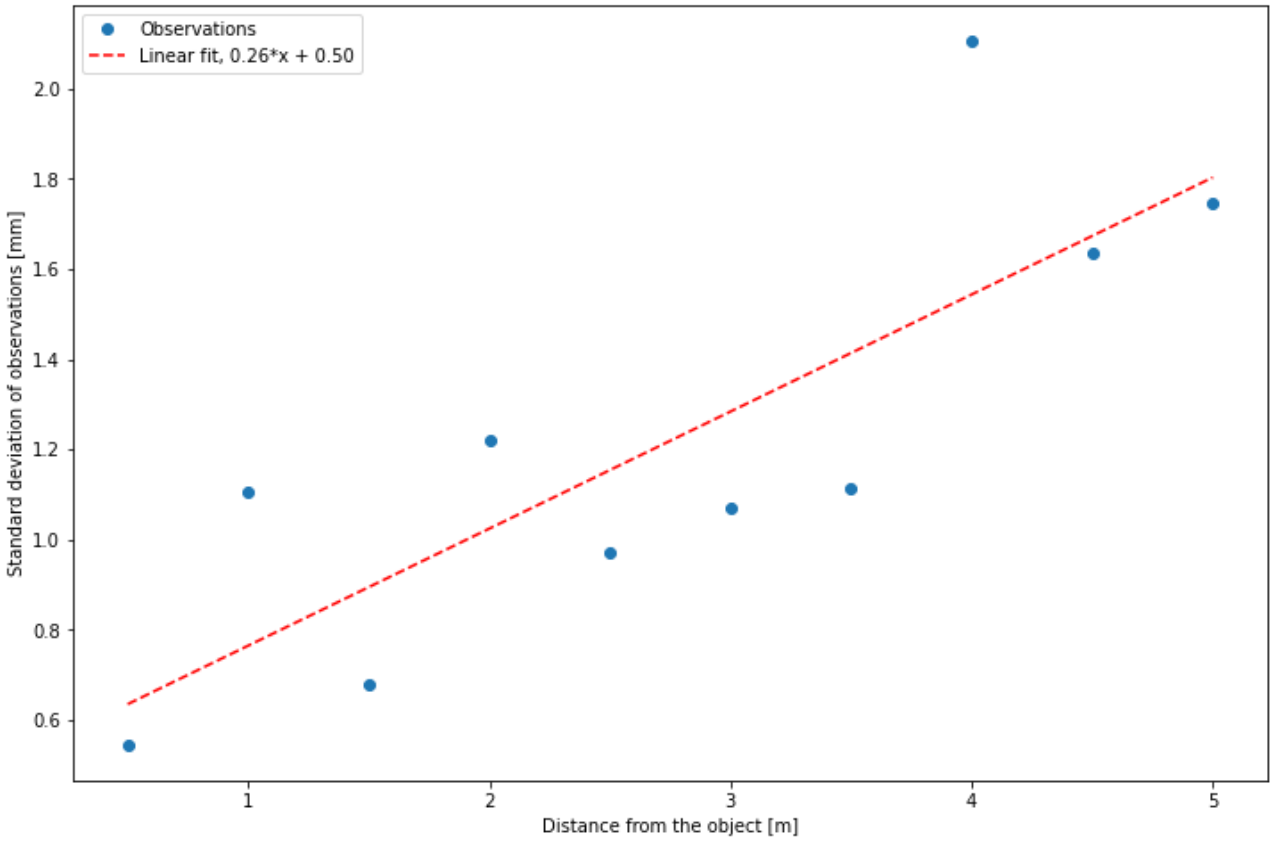
```
plt.close()
plt.bar(
    intervals,
    details["std"],
    width=0.1
    # marker='o',
    # ls='',
)
ax = plt.gca()
ax.set_xlabel("Distance from the target [m]")
ax.set_ylabel("Standard deviation of range measurements [mm]")
ax.set_ylim(ymin=0);
```



Evaluation of linear relationship between distance and standard deviation

In [ ]:

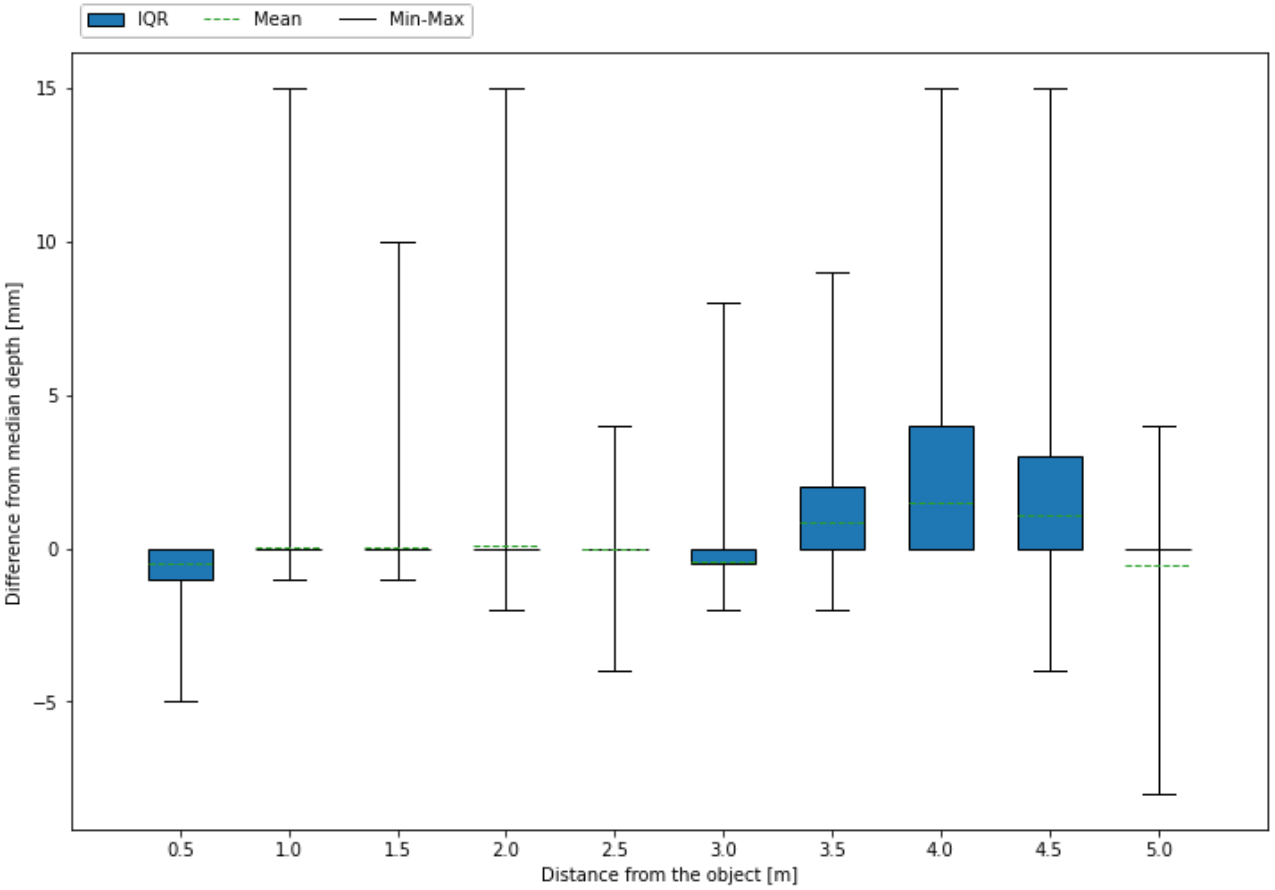
```
m, b, r_value, p_value, std_err = scipy.stats.linregress(
    intervals[0:10], details["std"][0:10]
)

plt.close()
plt.plot(intervals[0:10], details["std"][0:10], "o", label="Observations")

plt.plot(
    intervals[0:10],
    m * np.asarray(intervals[0:10]) + b,
    "r--",
    label="Linear fit, {:.2f}*x + {:.2f}".format(m, b),
)

ax = plt.gca()
ax.set_xlabel("Distance from the object [m]")
ax.set_ylabel("Standard deviation of observations [mm]")
ax.legend();
```



Boxplot of distributions against distance

In [ ]:

```
plt.close()
n_selected_intervals = 18
box = plt.boxplot(
    (
        range_measurements[0:10].transpose() - details["median"][0:10]
    ),  # [:,0:n_selected_intervals],
    whis=(0, 100),
    positions=intervals[0:10],  # [0:n_selected_intervals],
    widths=0.3,
    medianprops={"linewidth": 0},
    patch_artist=True,
    showmeans=True,
    meanline=True,
)

# plt.xticks(range(0,8),range(0,8))
ax = plt.gca()
ax.set_xlabel("Distance from the object [m]")
ax.set_ylabel("Difference from median depth [mm]")
ax.legend(
    [box["boxes"][0], box["means"][0], box["whiskers"][0]],
    ["IQR", "Mean", "Min-Max"],
    loc="lower left",
    bbox_to_anchor=(0.0, 1.01),
    ncol=3,
);
```



Reading and visualizing small scale measurements

In [ ]:

```
range_measurements = []
intervals = [
    0,
    0.01,
    0.02,
    0.03,
    0.04,
    0.05,
    0.06,
    0.07,
    0.08,
    0.09,
    0.1,
    0.12,
    0.14,
    0.16,
    0.18,
    0.2,
    0.22,
    0.24,
    0.26,
    0.28,
]

for folder in os.scandir(data_folder + "/macro/"):
    if not folder.is_dir():
        continue  # Neglect files, read subfolders only
    print(folder.path)

    depth_data, _, _ = read_data(folder.path)
    range_measurements.append(
        depth_data[60:540, 192 // 2, 256 // 2]
    )  # Removing the effect of shaking by leaving points away from the ends,
       # using only central pixel values

range_measurements = np.asarray(range_measurements)
```

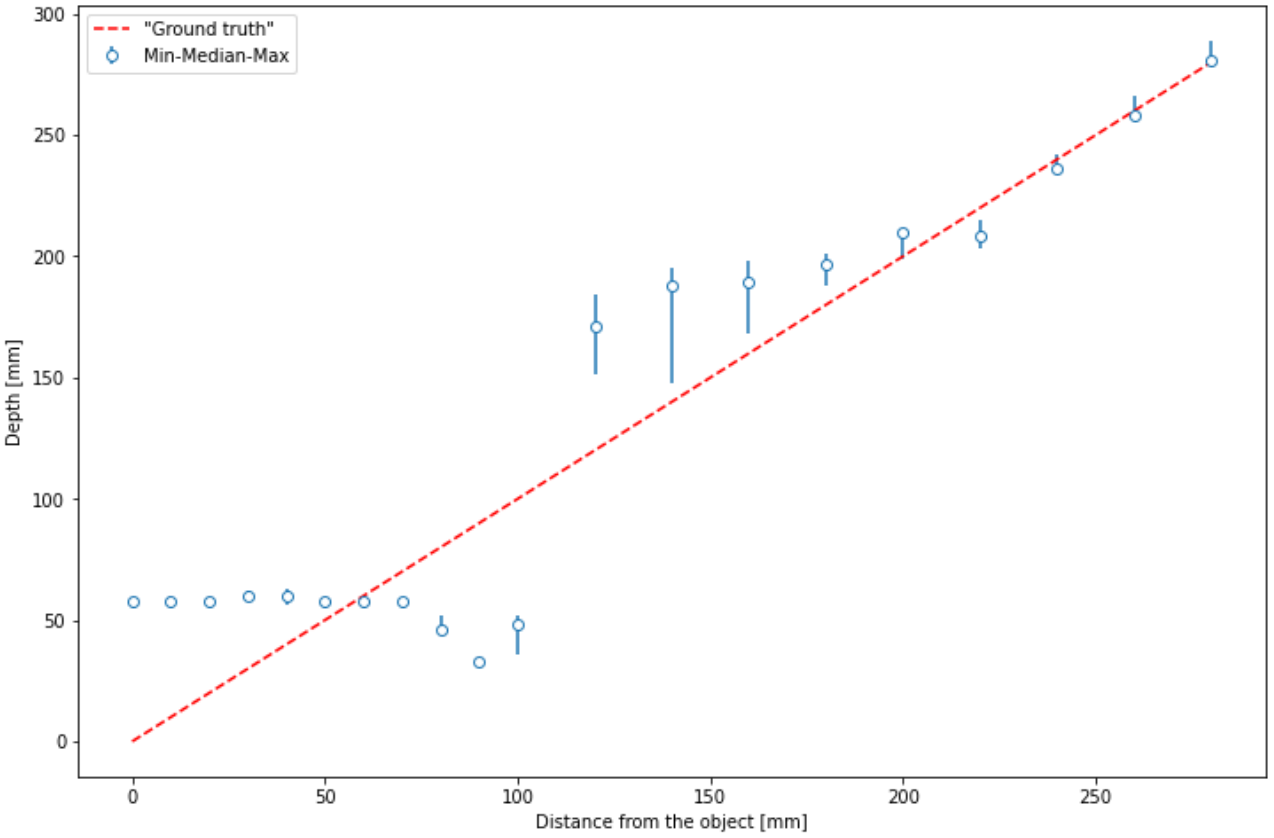
```
../Data/macro/000mm
../Data/macro/010mm
../Data/macro/020mm
../Data/macro/030mm
../Data/macro/040mm
../Data/macro/050mm
../Data/macro/060mm
../Data/macro/070mm
../Data/macro/080mm
../Data/macro/090mm
../Data/macro/100mm
../Data/macro/120mm
../Data/macro/140mm
../Data/macro/160mm
../Data/macro/180mm
../Data/macro/200mm
../Data/macro/220mm
../Data/macro/240mm
../Data/macro/260mm
../Data/macro/280mm
```

In [ ]:

```
details = dict(
    median=np.median(range_measurements, axis=1),
    mean=np.mean(range_measurements, axis=1),
    std=np.std(range_measurements, axis=1),
    min=np.min(range_measurements, axis=1),
    max=np.max(range_measurements, axis=1),
    upper=np.percentile(range_measurements, 75, axis=1),
    lower=np.percentile(range_measurements, 25, axis=1),
    iqr=scipy.stats.iqr(range_measurements, axis=1),
)
```

In [ ]:

```
plt.close()

minmax = [
    details["median"] - details["min"],
    details["max"] - details["median"],
]

plt.plot(
    [i * 1000 for i in intervals],
    [i * 1000 for i in intervals],
    color="r",
    ls="--",
    label='"Ground truth"',
)
plt.errorbar(
    [i * 1000 for i in intervals],
    details["median"],
    minmax,
    fmt="o",
    mfc="w",
    capthick=10,
    label="Min-Median-Max",
)

ax = plt.gca()
ax.set_xlabel("Distance from the object [mm]")
ax.set_ylabel("Depth [mm]")
ax.legend();
```



4.4 Depth measurement consistency over a depth map

This experiment pursues to illustrate how coherent the repeatability is along whole depth map. Both repeatability anomalies and systematic distortions are assessed

Anomalies in pixel standard deviation

Standard deviations are computed for each pixel individually, and illustrated in image format.

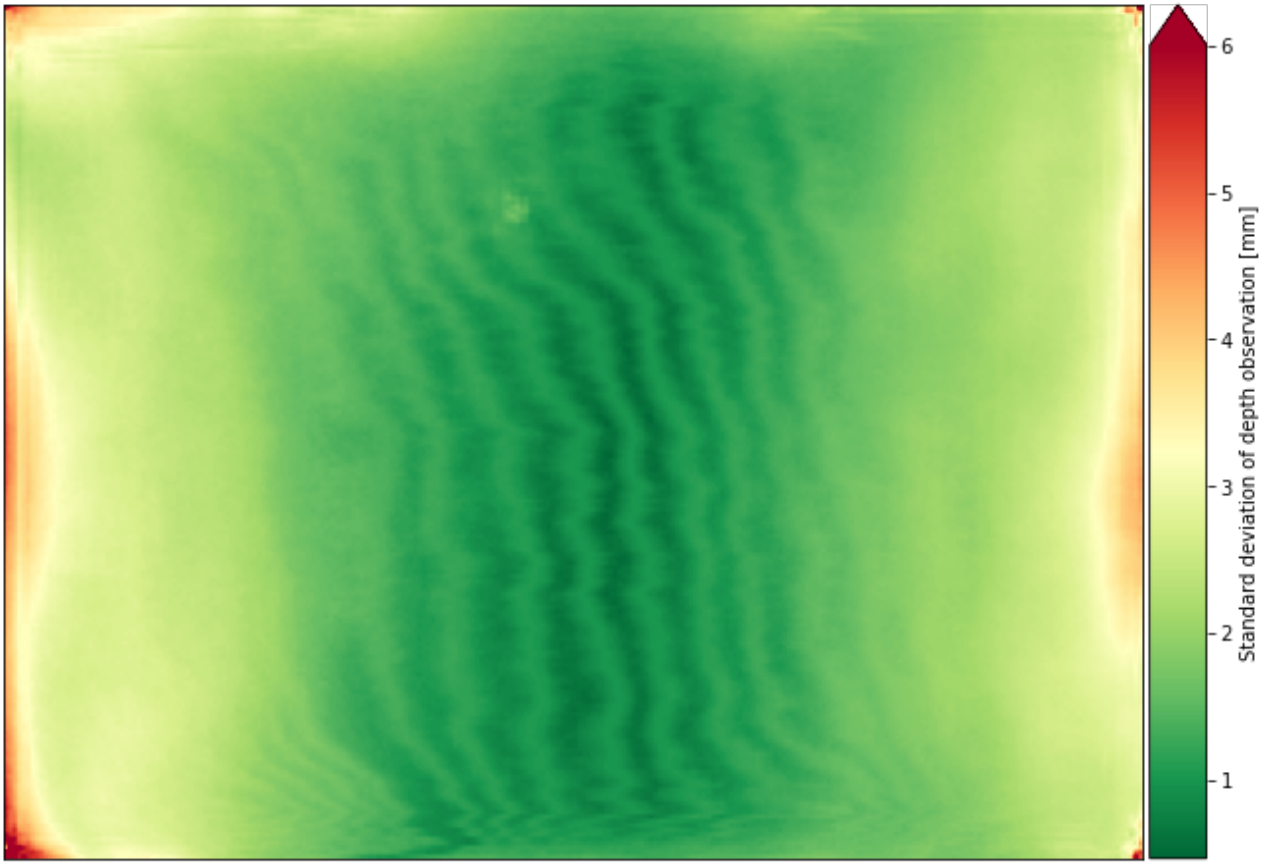
In [ ]:
```
depth_data, _, _ = read_data("../Data/plainWall_cameraOff")
depth_std = np.std(depth_data, axis=0)
depth_std.shape
```

Out[ ]:
```
(192, 256)
```

Displaying results in image format.

In [ ]:

```
plt.close()
std_fig = plt.figure()
std_ax = plt.subplot()
std_im = std_ax.imshow(depth_std, cmap="RdYlGn_r", vmax=6)
plt.tick_params(
    axis="both",
    which="both",
    bottom=False,
    labelbottom=False,
    left=False,
    labelleft=False,
)
std_ax.set_xlabel(
    ("Highest deviation value is {:.1f} mm and "
    "exceeds the color scale range").format(
        depth_std.max()
    ),
    loc="left",
    style="italic",
)

divider = make_axes_locatable(std_ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cbar = plt.colorbar(std_im, cax=cax, extend="max")
cbar.set_label("Standard deviation of depth observation [mm]")
```



Highest deviation value is 14.7 mm and exceeds the color scale range

Systematic distortions

This experiment pursues to illustrate magnitude of distortions within a depth map.

Create a point cloud from depth map stack

In [ ]:
```
depth_data, _, _ = read_data("../Data/plainWall_cameraOff")
intrinsics = read_intrinsics("../Data/plainWall_cameraOff/camera_matrix.csv")
median_depth = np.mean(depth_data, axis=0).astype("uint16")
pointcloud = o3d.geometry.PointCloud.create_from_depth_image(
    o3d.geometry.Image(median_depth), intrinsics
)
```

In [ ]:
```
o3d.visualization.draw_geometries(
    [pointcloud]
)  # Inspect the point cloud data interactively
```

Fit a plane with RANSAC to the data

In [ ]:
```
plane_model, _ = pointcloud.segment_plane(
    distance_threshold=0.01, ransac_n=10, num_iterations=1000
)

[a, b, c, d] = plane_model
print(f"Plane equation: {a:.2f}x + {b:.2f}y + {c:.6f}z + {d:.2f} = 0")

angle = np.arccos(c / np.sqrt(a ** 2 + b ** 2 + c ** 2)) * 180 / np.pi
print(f"Angle between the wall and the sensor: {angle:.3f} degrees")
```

```
Plane equation: 0.02x + -0.01y + 0.999864z + -2.01 = 0
Angle between the wall and the sensor: 0.944 degrees
```

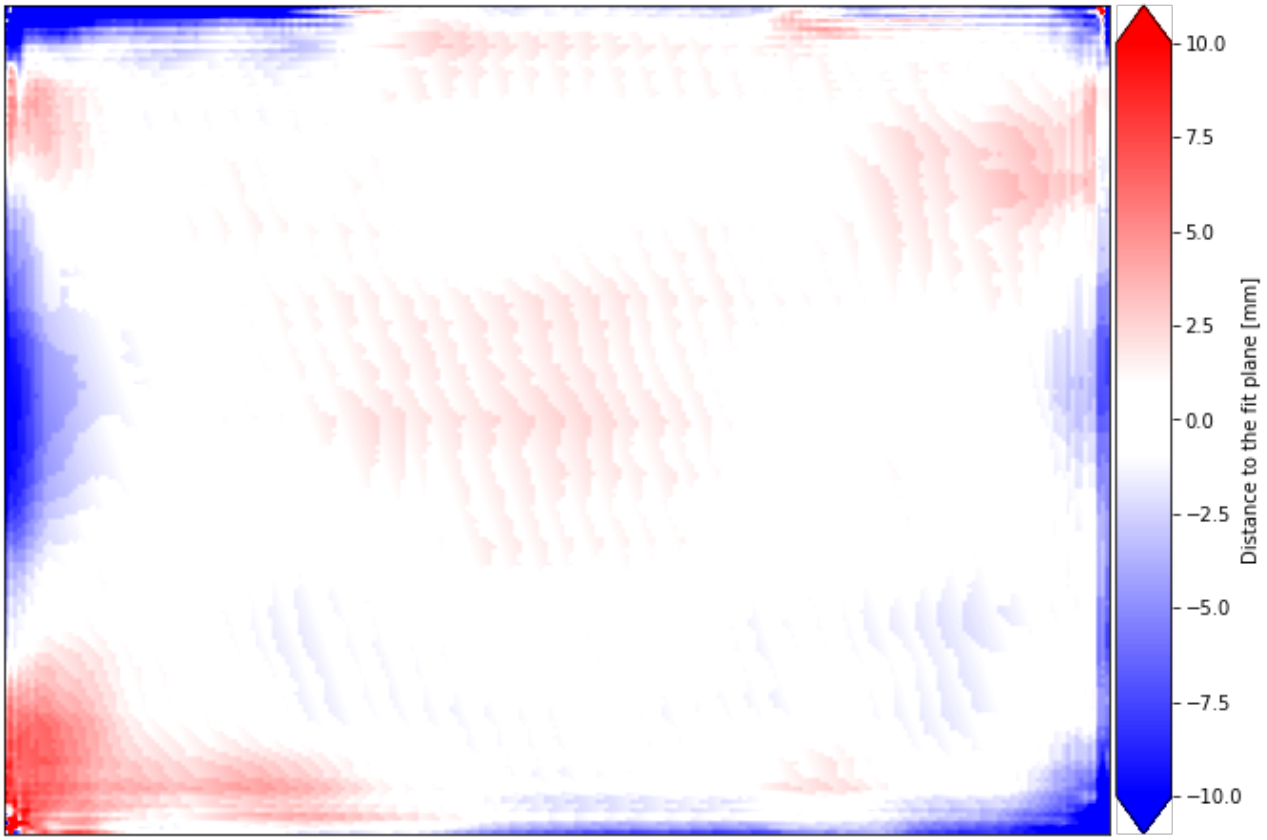
Define color map for visualization

In [ ]:
```
import matplotlib.colors

cdict = {
    "red": ((0.0, 0.0, 0.0), (0.45, 1.0, 1.0), (1.0, 1.0, 1.0)),
    "green": (
        (0.0, 0.0, 0.0),
        (0.45, 1.0, 1.0),
        (0.55, 1.0, 1.0),
        (1.0, 0.0, 0.0),
    ),
    "blue": ((0.0, 1.0, 1.0), (0.55, 1.0, 1.0), (1.0, 0.0, 0.0)),
}
cmap = matplotlib.colors.LinearSegmentedColormap("bWr", cdict)
```

For each point, compute the distance to the plane and visualize systematic differences in image format.

In [ ]:

```
plt.close()
distances = distances_to_plane(pointcloud.points, plane_model)
distances = distances.reshape((192, 256))
std_fig = plt.figure()
ax = plt.subplot()
im = plt.imshow(distances * 1000, cmap=cmap, vmin=-10, vmax=10)

plt.tick_params(
    axis="both",
    which="both",
    bottom=False,
    labelbottom=False,
    left=False,
    labelleft=False,
)
ax.set_xlabel(
    ("Highest distance to the plane is {:.1f} mm and "
    " exceeds the color scale range").format(
        (np.abs(distances) * 1000).max()
    ),
    loc="left",
    style="italic",
)

divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cbar = plt.colorbar(im, cax=cax, extend="both")
cbar.set_label("Distance to the fit plane [mm]", labelpad=0)
```



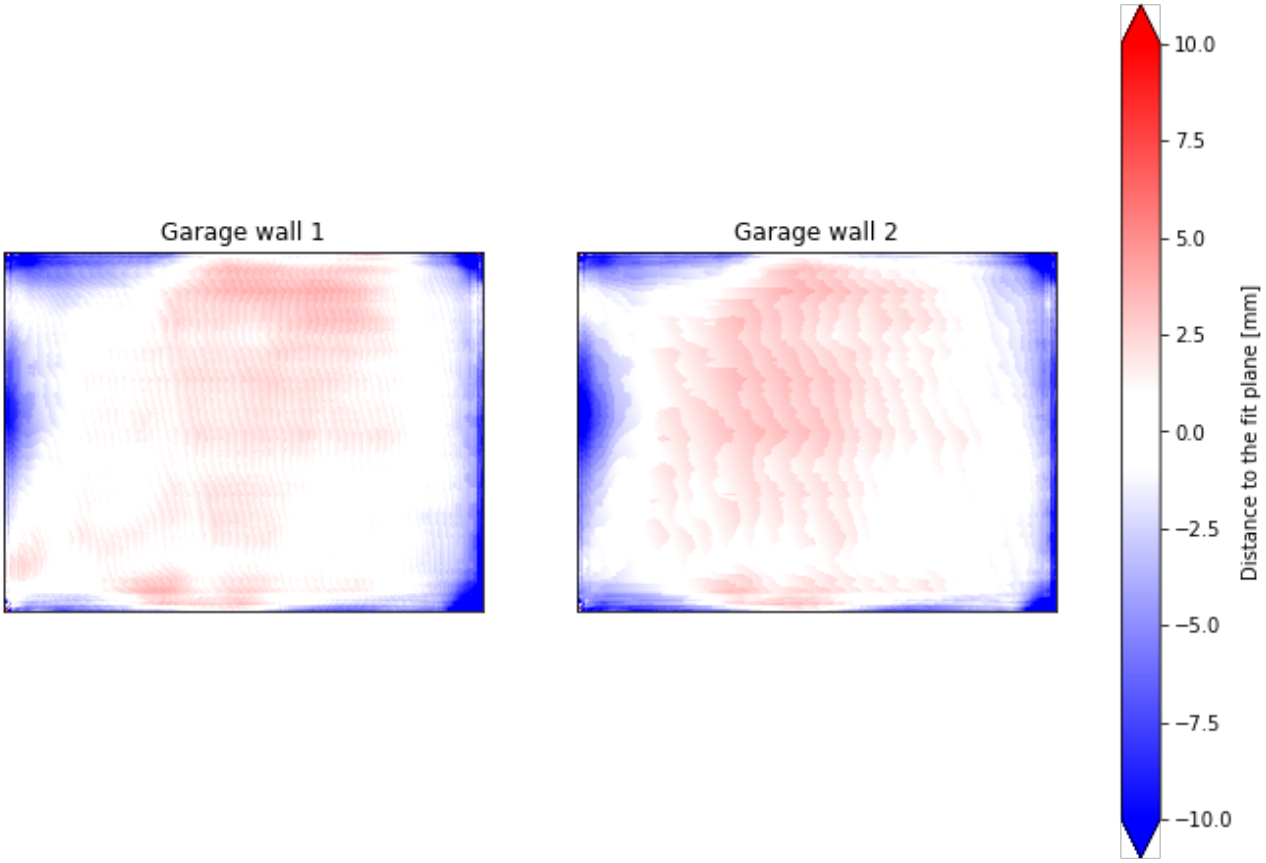
Highest distance to the plane is 92.3 mm and exceeds the color scale range

In [ ]:
```
blurred = cv.GaussianBlur(distances, (9, 9), 1)
ax.imshow(blurred, cmap=cmap, vmin=-0.01, vmax=0.01)
ax.set_xlabel(
    ("Highest distance to the plane is {:.1f} mm and "
    "exceeds the color scale range").format(
        (np.abs(blurred) * 1000).max()
    ),
    loc="left",
    style="italic",
);
```

Out[ ]:
```
Text(0, 17.200000000000003, 'Highest distance to the plane is 52.6 mm and exceeds the color scale range')
```

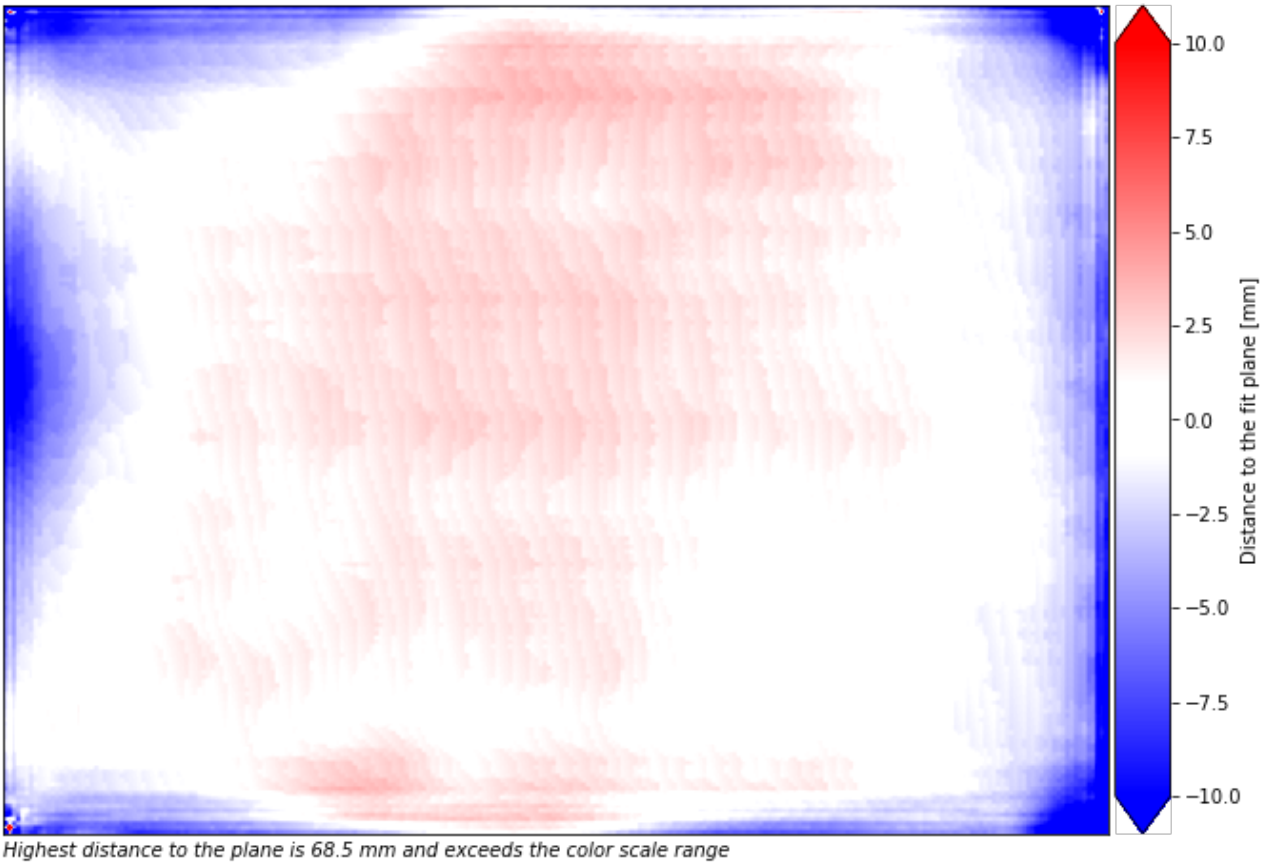
In [ ]:
```
distance_maps = []

dataset_folders = ["flat_1", "flat_2"]
dataset_names = ["Garage wall 1", "Garage wall 2"]

for folder, ax in zip(dataset_folders, axs.reshape(-1)):
    depth_data = read_depth_data("../Data/" + folder + "/depth/")
    intrinsics = read_intrinsics("../Data/" + folder + "/camera_matrix.csv")
    median_depth = np.mean(depth_data, axis=0).astype("uint16")
    pointcloud = o3d.geometry.PointCloud.create_from_depth_image(
        o3d.geometry.Image(median_depth), intrinsics
    )

    plane_model, _ = pointcloud.segment_plane(
        distance_threshold=0.01, ransac_n=10, num_iterations=1000
    )
    distances = distances_to_plane(pointcloud.points, plane_model)
    distances = distances.reshape((192, 256))
    distance_maps.append(distances)
```

In [ ]:
```
plt.close()
fig, axs = plt.subplots(ncols=2)
for image, name, ax in zip(distance_maps, dataset_names, axs.reshape(-1)):
    im = ax.imshow(image * 1000, cmap=cmap, vmin=-10, vmax=10)
    ax.tick_params(
        axis="both",
        which="both",
        bottom=False,
        labelbottom=False,
        left=False,
        labelleft=False,
    )
    ax.set_title(name)


cbar = fig.colorbar(im, ax=axs.ravel().tolist(), extend="both")
cbar.set_label("Distance to the fit plane [mm]")
```



In [ ]:

```
plt.close()
distances = np.asarray(distance_maps).mean(axis=0)
distances = cv.GaussianBlur(distances, (9, 9), 0.1)
std_fig = plt.figure()
ax = plt.subplot()
im = plt.imshow(distances * 1000, cmap=cmap, vmin=-10, vmax=10)

plt.tick_params(
    axis="both",
    which="both",
    bottom=False,
    labelbottom=False,
    left=False,
    labelleft=False,
)
ax.set_xlabel(
    ("Highest distance to the plane is {:.1f} mm and "
    "exceeds the color scale range").format(
        (np.abs(distances) * 1000).max()
    ),
    loc="left",
    style="italic",
)

divider = make_axes_locatable(ax)
cax = divider.append_axes("right", size="5%", pad=0.05)
cbar = plt.colorbar(im, cax=cax, extend="both")
cbar.set_label("Distance to the fit plane [mm]", labelpad=0)
```



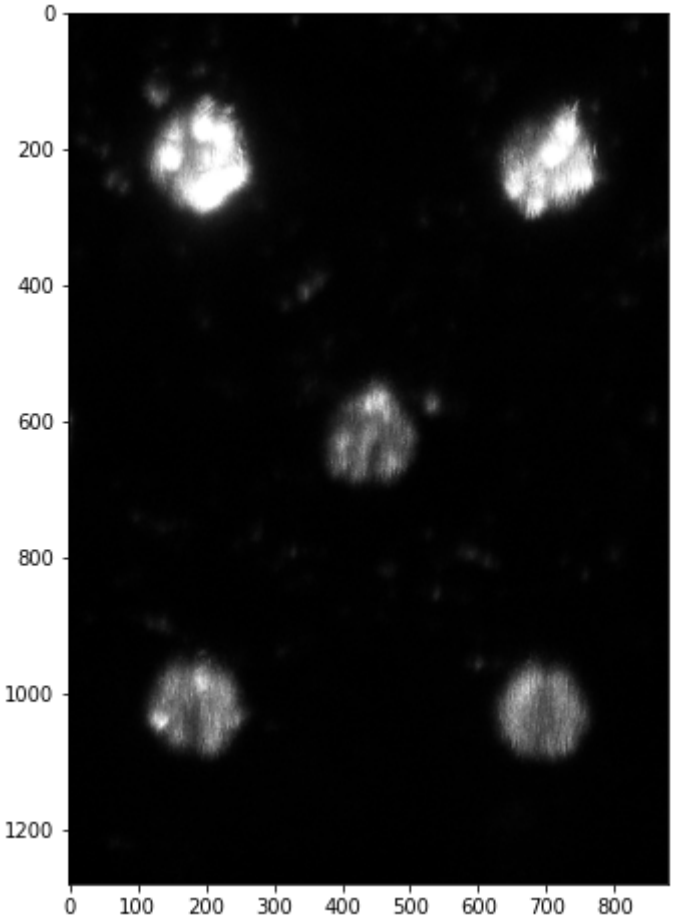
Highest distance to the plane is 68.5 mm and exceeds the color scale range

4.5 Testing with different object materials

Check crop area for further processing

In [ ]:

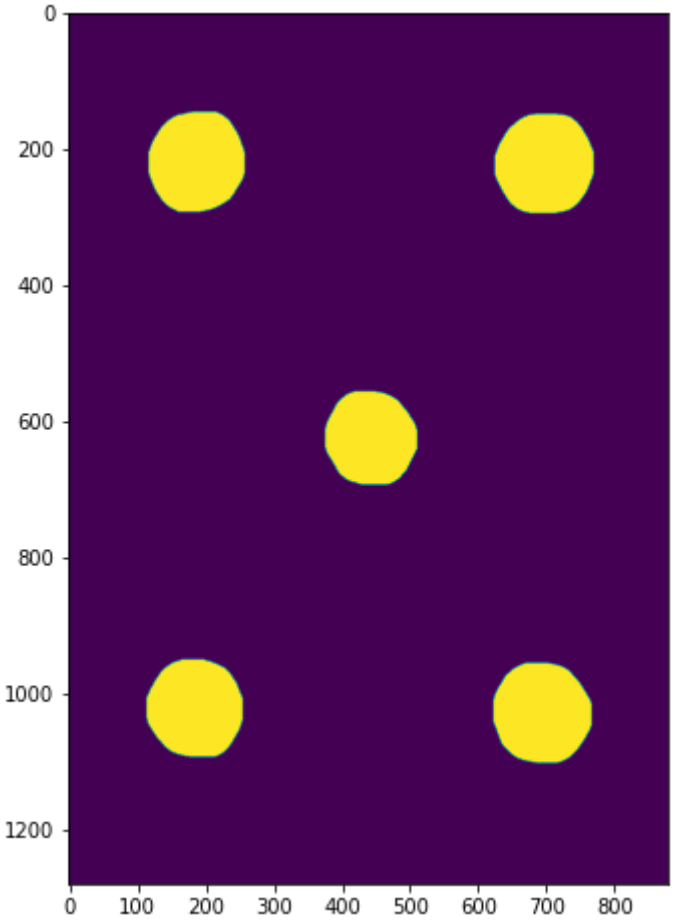
```
intensity_data = read_intensity_data(
    data_folder + "/different_materials/aluminium"
)
plt.close()
plt.imshow(intensity_data[520:1800, 1170:2050], cmap="gray");
```



Check segmentation quality

In [ ]:

```
intensity, segmented = extract_intensity(
    intensity_data[520:1800, 1170:2050], debug=True
)
plt.close()
plt.imshow(segmented > 0);
```



Read data and extract intensities

In [ ]:

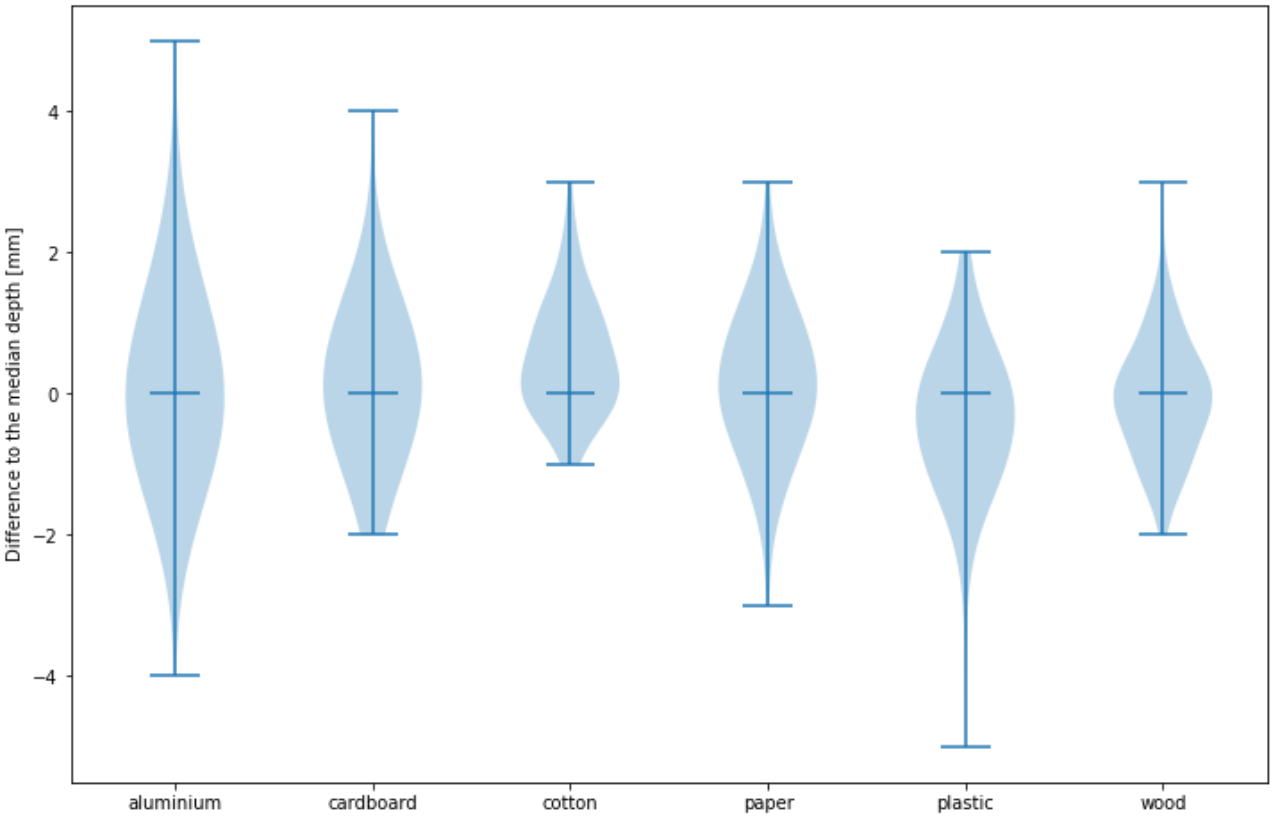
```
range_measurements = []
reflected_intensities = []
materials = []

for folder in os.scandir(data_folder + "/different_materials/"):
    if not folder.is_dir():
        continue  # Neglect files, read subfolders only
    print(folder.path)

    materials.append(folder.name)

    depth_data, _, _ = read_data(folder.path)
    range_measurements.append(
        depth_data[60:540, 192 // 2, 256 // 2]
    )  # Removing the effect of shaking by leaving points away from the ends,
       # using only central pixel values

    intensity_data = read_intensity_data(folder.path)
    reflected_intensities.append(
        extract_intensity(intensity_data[520:1800, 1170:2050])
    )  # Use AOI detected in previous step


range_measurements = np.asarray(range_measurements)
materials
```

```
../Data/different_materials/aluminium
../Data/different_materials/cardboard
../Data/different_materials/cotton
../Data/different_materials/paper
../Data/different_materials/plastic
../Data/different_materials/wood
```

Out[ ]:
```
['aluminium', 'cardboard', 'cotton', 'paper', 'plastic', 'wood']
```

Visualize distributions for each material

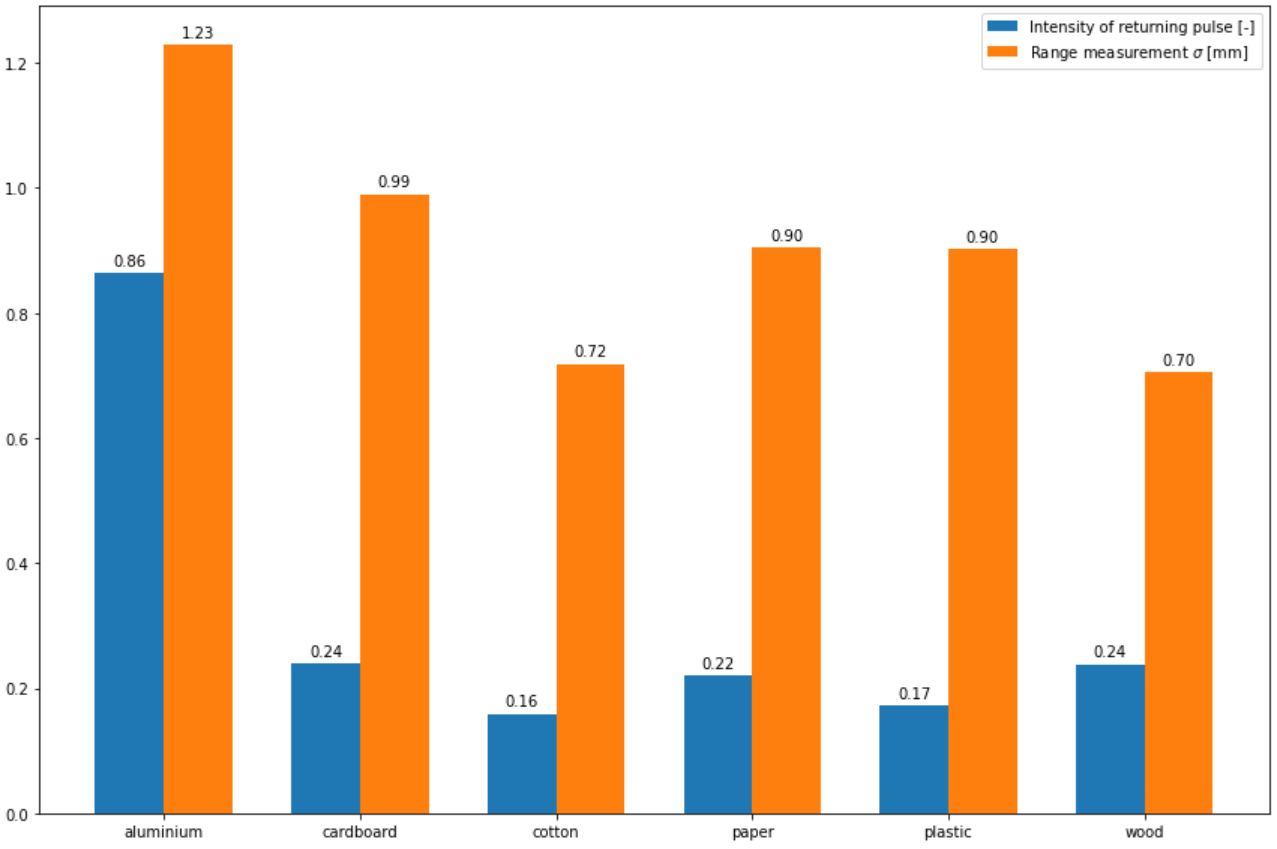
In [ ]:
```
plt.close()
ax = plt.subplot()
plt.violinplot(
    range_measurements.T-np.median(range_measurements, axis=1),
    bw_method=0.75,
    showmedians=True,
)
plt.ylabel("Difference to the median depth [mm]")
ax.set_xticks(np.arange(1,7))
ax.set_xticklabels(materials);
```



Visualize a comparison between intensities and accuracies

In [ ]:

```
plt.close()

width = 0.35  # the width of the bars
x = np.arange(len(materials))

fig, ax = plt.subplots()
rects1 = ax.bar(
    x - width / 2,
    np.asarray(reflected_intensities) / 255,
    width,
    label="Intensity of returning pulse [-]",
)
rects2 = ax.bar(
    x + width / 2,
    np.std(range_measurements, axis=1),
    width,
    label="Range measurement $\sigma$ [mm]",
)

ax.set_xticks(x)
ax.set_xticklabels(materials)
ax.legend()

ax.bar_label(rects1, padding=3, fmt="%.2f")
ax.bar_label(rects2, padding=3, fmt="%.2f")

fig.tight_layout()

plt.show()
```



Helper functions

For reading data

In [ ]:

```
import os
import numpy as np
import cv2
from PIL import Image
import open3d as o3d


def read_depth_data(folder="."):
    """
    A function for reading Stray Scanner depth maps
    and placing those to a numpy array.
        Input:
            folder: The path of the directory where depth maps are located
        Output:
            A numpy array of depth maps, shape (n of depth maps, height, width)
    """
    depth_data = []

    for fileName in os.listdir(folder):
        if not fileName.endswith(
            ".npy"
        ):  # Validating that input files are in correct format
            continue
        depth_data.append(np.load(folder + fileName))

    depth_data = np.asarray(depth_data)
    return depth_data


def read_confidence_data(folder="."):
    """
    A function for reading Stray Scanner confidence maps
    and placing those to a numpy array.
        Input:
            folder: The path of the directory where confidence maps are located
        Output:
            A numpy array of confidence maps,
            shape (n of confidence maps, height, width)
    """
    conf_data = []

    for file in os.listdir(folder):
        if not file.endswith(
            ".png"
        ):  # Validating that input files are in correct format
            continue
        conf_data.append(np.asarray(Image.open(folder + file)))

    conf_data = np.asarray(conf_data)
    return conf_data


def read_rgb_data(file="./rgb.mp4"):
    """
    A function for reading rgb video and placing frames to a numpy array.
    Note that with Stray Scanner versions up to 1.1 (at least) some frames
    are lost in the file creation process
    (so there might be offset between depth and rgb).
        Input:
            file: The path to the video file
        Output:
            A numpy array of confidence maps, shape
            (n of confidence maps, height, width)
    """
```

```
    rgb_data = np.empty(
        (nFrames, frameHeight, frameWidth, 3), np.dtype("uint8")
    )
    i = 0
    ret = True

    while i < nFrames and ret:
        ret, image = video.read()
        try:  # Handling errors caused by non-complete video files, as Stray
            # Scanner does not write all frames succesfully in version 1.1
            rgb_data[i] = image
        except:
            rgb_data[i] = 0
        i += 1

    video.release()
    return rgb_data


def read_data(
    folder=".", depth_maps=True, confidence_maps=False, rgb_frames=False
):
    """
    A function for reading Stray Scanner output data
    and placing it to numpy arrays.
        Input:
            folder:             String defining the data folder
            depth_maps:         Boolean defining whether to
                                read depth maps or not
            confidence_maps:    Boolean defining whether to
                                read confidence maps or not
            rgb_frames:         Boolean defining whether to
                                read rgb video or not
        Output:
            A tuple of numpy arrays containing depth data,
            confidence data and rgb data in respective order.
            If some pieces of the information are not included in processing,
            None value will be returned.
    """

    depth_data = None
    conf_data = None
    rgb_data = None

    if depth_maps:  # Reading depth maps and storing data to a numpy array
        depth_data = read_depth_data(folder + "/depth/")

    if confidence_maps:
        conf_data = read_confidence_data(folder + "/confidence/")

    if rgb_frames:
        rgb_data = read_rgb_data(folder + "/rgb.mp4")

    return depth_data, conf_data, rgb_data


def read_intensity_data(folder="."):
    """
    A function for reading JPG images and
    returning averaged frame as a numpy array.
```

```
            ".JPG"
        ):  # Validating that input files are in correct format
            continue
        intensity_data.append(np.asarray(Image.open(folder + "/" + file)))

    intensity_data = np.asarray(intensity_data)
    intensity_data = np.mean(intensity_data, axis=0)  # Averaging over frames
    intensity_data = np.mean(intensity_data, axis=2)  # Averaging over channels
    return intensity_data


def read_intrinsics(file="./camera_matrix.csv"):
    """
    A function for reading Stray Scanner camera intrinsics and
    returning information in open3d format.
        Input:
            file: String defining the csv-file
        Output:
            open3d pinhole camera intrinsics object
    """
    dm_width = 256
    dm_height = 192
    intrinsic_data = np.loadtxt(file, delimiter=",")
    scale = (
        dm_width / 1920
    )  # depth map / rgb frame scale ratio, same as for height (192px/1440px)
    intrinsics = dict(
        fx=intrinsic_data[0, 0] * scale,
        fy=intrinsic_data[1, 1] * scale,
        cx=intrinsic_data[0, 2] * scale,
        cy=intrinsic_data[1, 2] * scale,
    )

    intrinsics = o3d.camera.PinholeCameraIntrinsic(
        width=dm_width, height=dm_height, **intrinsics
    )

    return intrinsics
```

For processing data

In [ ]:

```python
import numpy as np
import cv2 as cv


def distances_to_plane(points, plane):
    """
    A function for computing orthogonal distance from point to plane
      Input:
        points: A numpy array containing Euclidean coordinates of points
        plane:  A numpy array containing plane model parameters [a, b, c, d]
      Output:
        A numpy array holding the distance value for each point
    """
    h_points = np.hstack(
        (points, np.ones((len(points), 1)))
    )  # Expressing points in homogenous coordinates
    distances = np.dot(h_points, plane) / np.linalg.norm(plane[0:3])
    return distances


def extract_intensity(intensity_image, debug=False):
    """
    A function for segmenting laser spots from IR camera images and
    returning averaged intensity value.
      Input:
        intensity_image:    A numpy array holding IR images of
                            returning spot intensities
        debug:              Boolean value for including segmentation
                            result to the output
      Output:
        Averaged intensity of the laser spots. If debugging is true,
        returns a tuple having also the segmented image
    """
    intensity_image = cv.GaussianBlur(intensity_image, (25, 25), 0)

    # Segmenting individual laser spots
    illuminated = intensity_image > 10  # Thresholding intensity image
    kernel = np.ones((15, 15), np.uint8)
    illuminated = cv.morphologyEx(
        illuminated.astype("uint8"), cv.MORPH_OPEN, kernel, iterations=2
    )
    illuminated = cv.morphologyEx(
        illuminated.astype("uint8"), cv.MORPH_CLOSE, kernel, iterations=2
    )

    # Extracting laser intensities from spots
    n_labels, labels, _, _ = cv.connectedComponentsWithStats(
        illuminated
    )  # Detecting individual illumination points

    intensities = []
    for i in range(1, n_labels):  # Finding maximum intensity for each label
        intensities.append(intensity_image[labels == i].max())

    assert (
        len(intensities) == 5
    )  # Ensure that the pattern is segmented properly,
    # i.e., all spots are detected and none is left out of process
    if debug == True:
        return np.mean(intensities), labels

    return np.mean(intensities)  # Default return
```