Modeling Spatial Uncertainty for the iPad Pro Depth Sensor

ANTONIO ZEA
UWE D. HANEBECK

Depth sensors, once exclusively found in research laboratories, are quickly becoming ubiquitous in the mass market. After Apple's introduction of the iPad Pro 2020 with an integrated light detection and ranging (LIDAR) sensor, now even tablets and smartphones are capable of obtaining accurate 3-D information from their environments. This, in turn, increases the reach of applications from technical fields, such as SLAM, object tracking, and object classification, which can now be downloaded on millions of hand-held devices with a couple of taps. This motivates an analysis of the capabilities, strengths, and weaknesses of these depth streams. In this paper, we present a study of the spatial uncertainties of the iPad Pro 2021 depth sensor. First, we describe the hardware used by the device, and provide an overview of the machine learning algorithm that fuses information from the LIDAR sensor with color data to produce a depth image. Then, we analyze the accuracy and precision of the measured depth values, while giving attention to the resulting temporal and spatial correlations. Another important topic of discussion are the tradeoffs involved in the extrapolations that the depth system implements, such as how curvatures change at different distances. In order to establish a reference baseline, we also compare the obtained results to another widely known time-of-flight sensor, the Microsoft Kinect 2.

Manuscript received January 18, 2022; revised August 10, 2022; released for publication March 20, 2023

Antonio Zea and Uwe D. Hanebeck are with the Intelligent Sensor-Actuator-Systems Laboratory (ISAS), Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology (KIT), 76131 Germany (e-mail: antonio.zea@kit.edu, uwe.hanebeck@ieee.org).

Refereeing of this contribution was handled by Florian Meyer.

This work was supported by the German Federal Ministry of Education and Research through ROBDEKON Project under Grant 13N14675.

I. INTRODUCTION

Depth sensors have been used for several decades in a wide variety of fields, ranging from robotics to computer-aided design (CAD), medicine, entertainment, and even the arts. During this time, multiple depth sensing technologies have been developed based on different operating principles. For example, visual data, such as RGB or grayscale images, can be processed to determine disparities at given points, which in turn can be used to reconstruct depth information. Recent advances in machine learning (ML) [7], [10], [11] can even extrapolate depth from a static image by filling in missing information from previously trained scenes. Light detection and ranging (LIDAR) sensors measure depth by projecting a (usually rotating) laser pulse onto a scene and calculating the time it takes for the pulse to return. In time-of-flight (ToF) cameras, a specialization of this technology, the laser pulse is split into a dense array of thousands of points, measuring an entire scene in a single scan and achieving a resolution and frame rate comparable to small RGB cameras.

Until about a decade ago, depth sensors in general were out of reach of the mass market due to their price. This changed significantly when Microsoft introduced the Kinect in 2010, an affordable depth camera (150 Euro) based on structured light. Originally designed as a body tracker for the Xbox 360, it was quickly adopted everywhere from robotics [19] to metrology [22] and therapeutics [16]. Since then, depth sensors have become ubiquitous in the research and hobbyist communities, with newer generations increasing accuracy and robustness while reducing their size and price. However, they have generally remained separate stand-alone devices. This stands in contrast to RGB cameras, most of which are now integrated into PCs, laptops, and mobile devices.

Among the first commercial attempts to integrate depth sensors into mobile devices were Lenovo's Phab 2 PRO in conjunction with Google's Tango platform [20] in 2016. Intended applications included immersive augmented reality (AR) experiences, scene reconstruction, and indoor tracking. However, there were few apps capable of exploiting these capabilities, leading to low interest from consumers. In turn, Tango was discontinued in 2017 and replaced with ARCore [13], which extracts depth information from RGB images and ML postprocessing. Since then, other smartphone manufacturers have made integration attempts, such as the Samsung Galaxy S20 with a ToF sensor, but it was discontinued for the release of the S22. Microsoft also integrated ToF sensors into its Hololens devices for AR (both first- and second-gen), but since 2019, they have made no concrete announcements about a third-gen device. These ebbs and flows of depth sensor integration are a consequence of a developer chicken-and-egg problem: if there are no apps, users will not buy the devices, but if users are not interested, then devel-

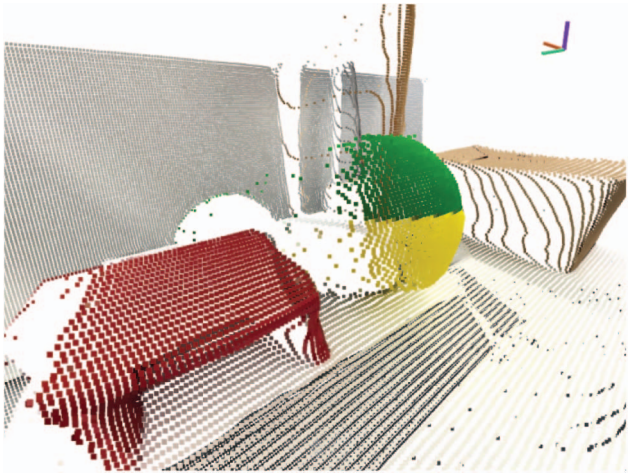


Fig. 1. Example of reconstructed point cloud for a table scene with objects, captured by an iPad Pro 2021 tablet.

opers have no reason to implement apps in the first place.

However, this situation is quickly changing with the introduction of platforms, such as Apple's ARKit [3], and the entrance of new competitors, such as Magic Leap and Facebook (now Meta). The newly introduced depth sensors of Apple are particularly noteworthy. In 2017, they announced a depth sensing technology on the iPhone X called "TrueDepth," which uses an infrared dot pattern similar to the first generation Kinect and has a range of up to 40 cm. Later, in 2020, they released a ToF sensor on the back of the iPad Pro with range of up to 5 m (see Fig. 1 for an example scene). Currently, all of Apple's mid and high-tier smartphones and tablets come with a depth sensor, and given the significant market share of their mobile devices, it is likely that their depth sensing technology will be the most widely used among nontechnical users in the near future. In this brief time, the iPad and iPhone LIDARs have already found versatile applications outside of the intended target of AR, ranging from forestry [12], [29], [31] and architecture [27] to heritage documentation [23] and geology [18], and even veterinary medicine [21]. This motivates an in-depth analysis of these sensors, in order to determine their usability in fields, such as localization and tracking. Similar depth sensors, such as the Microsoft Kinect (first and second-gen), have been extensively studied in literature [5], [8], [26], [32]. The TrueDepth system has already received some attention, for example, in [4] and [30], where the iPad is contrasted with an industrial scanner. The measurement accuracy of the iPhone 12 LIDAR [17] and the iPad Pro 2020 LIDAR [23], [30] has also been evaluated, but in the context of 3-D scanning and static reconstruction. However, we have not been able to find works that deal with quantitative models of spatial uncertainties for the iPad LIDAR.

The rest of this paper is organized as follows. Section II contains a description of the depth sensor, including the hardware and the streams provided by the API. Section III introduces a quantitative analysis of the measurements provided by the depth streams and their uncertainties. Finally, Section IV concludes this article. Throughout the paper, we will compare the iPad depth sensor to the well-known Kinect 2 device, which will serve as a baseline for its capabilities. However, we emphasize that these comparisons are merely illustrative, as both devices have different capabilities and are not aimed at the same range of applications.

II. IPAD DEPTH SYSTEM

The iPad depth sensor was introduced with the iPad Pro 2020 [2], the first Apple device to use a ToF sensor. It works by fusing depth and color streams together using ML algorithms. Its main application is in ARKit, Apple's AR platform, where it is employed for depth occlusion, scene understanding, and the detection, segmentation, and tracking of objects and humans. Fig. 2 shows an example of the data streams provided by the device, which when converted into a point cloud produced the image seen in Fig. 1.

The LIDAR sensor is located on the back (or rear) of the tablet, i.e., the side pointing away from the user (see Fig. 3). The hardware appears to be identical in both the 11 and the 12.9 in variants of the iPad Pro 2020 and the iPad Pro 2021, and a related study [17] found no difference with the iPhone 12. Also note that the front of the device (the display side) provides another depth system called "TrueDepth," based on structured light. Its main use is face recognition for authentication (FaceID) and face tracking, employed, for example, in Snapchat filters and "lenses." Both depth systems also differ in their operating range. While TrueDepth works best at a distance of at most 40–50 cm [30], the LIDAR sensor can measure walls up to 5 m away with moderate accuracy. This distinction also affects where they can be used. For example, while detecting small objects on a table is better suited for the TrueDepth system, a localization application in a large room would prefer the LIDAR data instead. Note that the TrueDepth system will *not* be considered in this work.

A. Sensors and Data Streams

We start by introducing the sensors on the back of the tablet, as shown in Fig. 3. On the top left is a wide-angle RGB camera, which from our experiments does not appear to be used in the depth system. On the top right is the standard RGB camera, with a smaller field of view but higher resolution. On the bottom left is the infrared flood illuminator, which ensures that the scene has an appropriate amount of light for the LIDAR system. On the bottom right is the flashlight, also not used by the depth system. Finally, the ToF sensor at the center consists of two submodules [28]. On the one hand, a vertical-cavity surface-emitting laser (VCSEL) diode is in charge of emitting a laser pulse which is split by

(a) Color image (b) Depth image (in colormap) (c) Confidence image (in colormap)

Fig. 2. Example capture of the scene from Fig. 1 showing the three video streams of the iPad depth system: the color image (1920 × 1440 px, RGB), the depth image (256 × 192 px, 32-bit float array), and the confidence image with 256 × 192 pixels, 1 byte/px, and three possible discrete values: low (black), middle (brown), and high (orange) confidence. (a) Color image. (b) Depth image (in colormap). (c) Confidence image (in colormap).

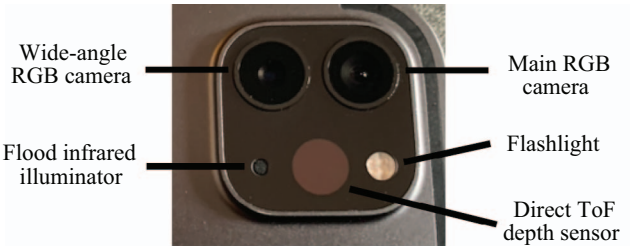


Fig. 3. Sensors on the back of the iPad Pro 2021. Width and height of the panel are around 27 mm.

a lens into 3 × 3 blocks with 8 × 8 dots each [1], [17], which are then projected onto the scene. A sketch of the pattern can be seen in the left-hand side of Fig. 4. On the other hand, a CMOS sensor captures the reflected light and calculates the distance between the device and each of those dots. Note that this suggests that each depth frame is interpolated from only $24 \times 24 = 9 \times 64 = 576$ measurements [1].

The depth data from these pulses are not directly available from the software library. Instead, the ARKit platform processes this information internally and fuses it with the color stream to produce a depth estimate. The result is a stream of three images at 60 frames/s (Fig. 4, right-hand side): a color image, a "scene" depth image, and a confidence image (see Fig. 2). For the sake

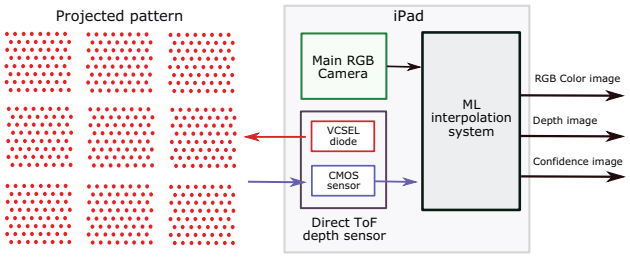
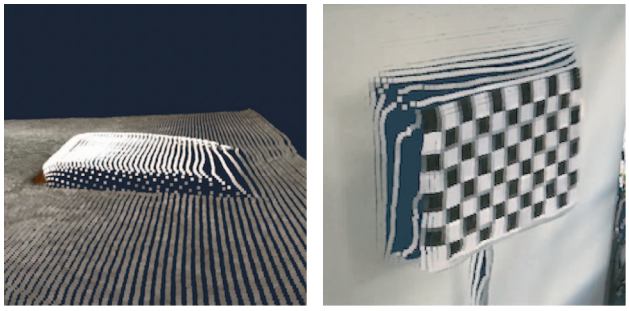


Fig. 4. Sketch of the depth capture process: a pattern of 576 dots is projected onto the scene, their distances are measured by the depth sensor, and the result is fused with the data from the RGB camera. The result are three streams: the color image, the depth image, and the confidence image.

of completeness, we note that the depth API also provides other streams, such as a real-time mesh reconstruction and a point cloud of RGB feature points, which will not be considered in this work. The values in this paper were captured using ARKit 5 and iOS 15.2.

The color image [see Fig. 2(a)] is an uncompressed packed RGB stream with a resolution of 1920×1440 pixels (px) and 24 bits/px. The original YUV image is also accessible if desired. The depth image [see Fig. 2(b)] has a resolution of 256 × 192 px, and each pixel contains a 32-b float describing the depth in meters at that position. The depth image is already registered to the color image and has the same aspect ratio of 4 to 3. The field of view of both images is about 60° horizontal and 48° vertical, with slight variations between devices. Finally, ARKit also provides a confidence map [see Fig. 2(c)] which determines how accurate the depth value of each pixel is. It has the same resolution as the depth image, but each pixel has an 8-bit integer value, which can be either 0 (low), 1 (medium), or 2 (high). Unlike sensors, such as the Microsoft Kinect, an invalid measurement is not encoded with a depth value of 0. Instead, the corresponding confidence is set to 0, and the depth is extrapolated based on surrounding depth values and semantic cues from the color image. In Fig. 2, a confidence of 0 can be seen around the edges of the table, or at the image borders. However, these gaps are not evident when looking at the depth image on its own.

The three images in Fig. 2 correspond to the point cloud that was shown in Fig. 1. Here, the large amount of "fringe points" (also known as "flying pixels") at the borders are clearly visible, as a result of the relatively low LIDAR resolution. Looking at the brown cardboard box at the back, it is also clear that planar surfaces are not necessarily shown flat, and that 90^o corners are not usually preserved. More interestingly, we observe that the measurement noise is strongly spatially correlated but not temporally correlated. In other words, unlike depth sensors, such as the Microsoft Kinect 2, where each measurement moves back and forth independently from its neighbors, here we observe entire sur-



(a) Box with height 2 cm (b) Chessboard at 20 cm from a wall

Fig. 5. Illustration of how the iPad observes objects with steep depth changes. (a) Box with height 2 cm. (b) Chessboard at 20 cm from a wall.

faces appearing to move and deform "coherently" each frame, but in slightly different ways. This tendency can be appreciated, for example, in the fringe points of the tall brown cylinder, which form different curved lines each frame. These correlations are introduced by the interpolation system, which will be described in Section II-B.

B. ML Interpolation System

The ML interpolation system is in charge of filling the gaps between the sparse depth measurements by fusing data from multiple streams, in particular from the LIDAR and RGB sensors. A sketch of its workings can be found in the patent description [33]. Unfortunately, as of iOS 15, there is no way to turn it OFF and obtain the raw data. However, given that it has a significant effect on the provided measurements, it is of interest to describe qualitatively how it works and what it does.

Generally speaking, the iPad depth system acts by aggregating measurements into coherent surfaces and "smoothing" out sudden changes in depth. This can be seen in Fig. 5(a), where the back of the white box, being observed from 1 m away, merges into the floor. The capability of the iPad to discern depth discontinuities is reduced by the sensor distance, as can be seen in Fig. 5(b) with the sensor being 4 m away, where the chessboard "melts" into the wall. These smoothed out measurements can usually be identified by their confidence level of 1 (medium) or 0 (low), as shown in Fig. 2(c) in dark brown and black, allowing them to be filtered out.

As mentioned before, pixel positions with no valid depth are denoted with a confidence level of 0. Invalid measurements can happen for several reasons, such as a reflective or bright material, a drastic change in depth, or the depth being outside of the operative range. Very narrow objects will also fail to be detected, especially if their size is less than of about 6% of the image (i.e., 15 px for the depth image). This value is probably related to the distance between LIDAR dots, i.e., 256 px/row or 24 dots/row $\approx$ 11 px. Similarly, detection will also fail if the depth changes quickly at the image edges [see Fig. 2(c)]. For these regions, the depth image will not show invalid values. Instead, the ML interpolation system will fill in the gaps by "guessing" which depths belong there based on data from the color image.

As the ML interpolation combines data from both the color and depth images, it is of interest to see how they work when one data stream is unavailable, for example, by covering one camera with tape or a piece of paper. When the color stream is absent, the resulting depth image appears extremely blurred, lacking sharp corners and with significant sections of the confidence image with values of middle or low. When the depth sensor is covered, the entire confidence image has a value of low, and the depth inference from color is applied to the entire image. This can produce interesting results, such as seen in Fig. 6. The setup is an iPad tablet 20 cm away from a flat monitor that is showing an image of a rendered cube on a plane. Fig. 6(a) presents the RGB capture from the color camera. Fig. 6(b) illustrates the resulting point cloud with the depth and color sensors active. However, if we cover the color camera, the depth inference generates a scene with a cube at a distance of 2 m [see Fig. 6(c)]. As with the fringe points, these reconstructed depths possess spatial correlations but lack temporal correlation, and thus, will change drastically between frames. Note that these reconstructions do not only appear if the whole depth sensor is covered. For example, a shiny, reflective object somewhere in a scene (such as a laminated picture) can produce the same effect.

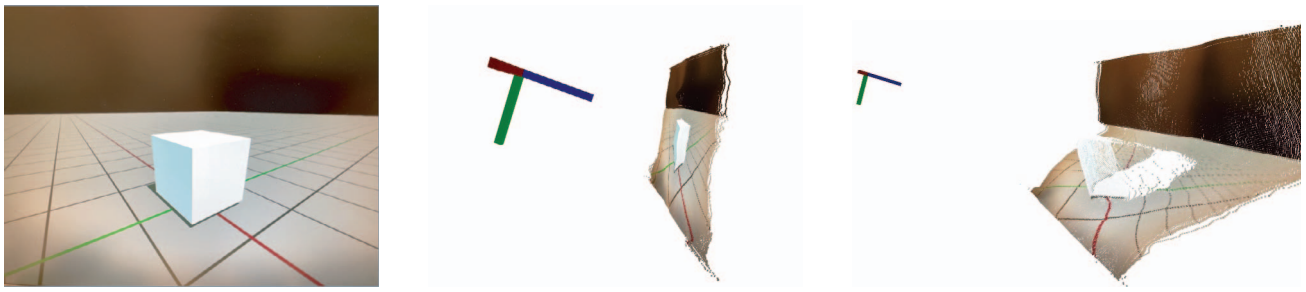
III. MODELING SPATIAL UNCERTAINTY

In this section, we will present a quantitative analysis of the uncertainties in the iPad depth system. First, we start by analyzing the depth discretization, which tells us the range of values that the system can provide. Then, we will measure the measurement bias, that is, the difference between the measured and the real depths. After this, we will focus on stochastic uncertainties and establish a measure for the correlations (in time and space) between measurements. Finally, we will present an analysis of how the measured curvature degrades as a function of distance.

A. Discretization

An important aspect to determine the quality of the depth stream is to see which depth values can be represented in the first place. The Kinect 1, for example, can only produce 2048 depth values spread out over the operating range of 0 to 8 m. The Kinect 2 has a much higher resolution, but as the depths are provided in millimeters as 16-b unsigned integers the distance between measurements are necessarily multiples of 1 mm. In contrast to these sensors, the resolution of the iPad depth system depends on the depth range.

Fig. 7 shows the quantization (also known as discretization) of the depth values, i.e., the distance between one value and the next, in function of the depth. This dis-



(a) Monitor showing a virtual scene (b) Point cloud with color and depth active (c) Inferred point cloud from color image

Fig. 6. Example of ML depth inference where the iPad observes a computer monitor from a closeup distance of 20 cm. When the depth sensor is covered, the iPad depth system takes the color image and extrapolates it into a new scene where the cube sits at a distance of 2 m. (a) Monitor showing a virtual scene. (b) Point cloud with color and depth active. (c) Inferred point cloud from color image.

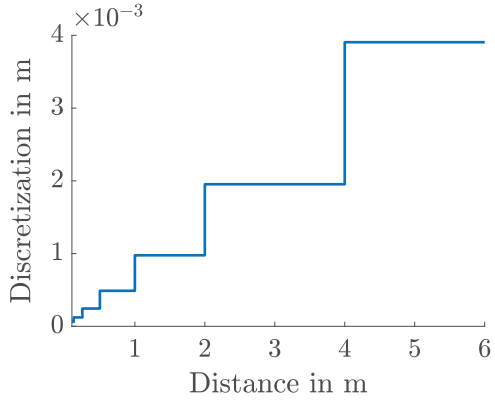


Fig. 7. Discretization of depth values, i.e., the distance between a value and the next possible one, depending on the distance.

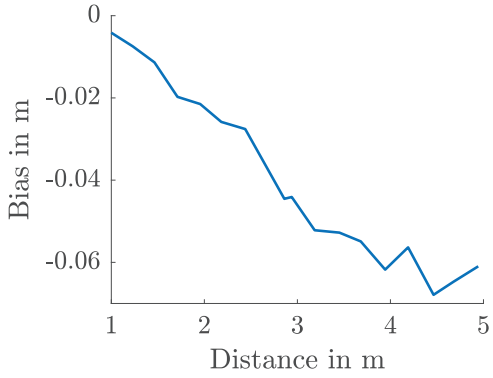


Fig. 8. Difference between the ground truth depths and the measured depths at difference distances. The negative bias means that the measured values are smaller than the ground truth.

cretization appears to be structured so that, between one power of two and the next (in meters), there are 1024 values. Thus, between 1 and 2 m we will see values at spaces of 1/1024 m, or around 0.977 mm. However, between 2 m and 4 m, the space will be 1/512 m instead, i.e., twice as large. Thus, the lower the resolution becomes, the larger the depth value is.

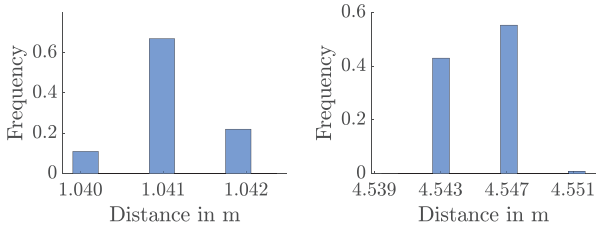
The range of possible values is difficult to measure. Any object farther than 5 m will have its measurements automatically marked as having confidence 0, and thus, the received depths will originate from the depth inference system and not from the LIDAR. Still, in our experiments, we have not observed a depth value above 8 m. Values below 10 cm will similarly be marked as low or medium confidence, yielding "reconstructed" depths that may have little relation to the actual physical distance.

B. Measurement Bias

As usual in real-life sensors, the measured depth values are not the true depths, as the process of capturing the data introduces errors, which depend on many factors, such as the pixel position, material, angle of reflection, temperature, and others. In order to keep the model simple, we can divide these errors into two additive components: a fixed offset (bias) and a zero-mean stochastic noise term. Both terms depend on the depth from which the measurement originated.

In this section, we will focus on estimating how the bias behaves at different distances. To achieve this, we captured 150 frames of a paper chessboard [70 × 49 cm, see Fig. 5(b)] at different distances, ranging from 1 to 5 m. The ground truth depth was obtained by detecting the chessboard in the color stream, estimating the board plane using MATLAB's implementation of the *solvePnP* algorithm, and finally calculating the depth that corresponded to the plane center. For the measured depth, we considered a square 5 × 5 px window around the center of the detected chessboard, and then calculated the average of all values for all frames. All of the considered points have a confidence of 2 (high). Fig. 8 shows the results.

It can be seen that the bias was always negative, that is, the iPad tells us that the object is closer than it really is. The bias also appears to increase linearly, but still remaining with 1% to 2% of the ground truth. However, this model stops being reliable at about 4 m, given the tendency of the chessboard surface to "melt" into the back wall at large distances. In these cases, the average depth can change significantly depending on where on the chessboard the window is located. After 5 m, the depth inference system kicks in, causing the obtained values to bear little relation to the real depths.



(a) Depth distribution at 1 m (b) Depth distribution at 4.5 m

Fig. 9. Representative probability distributions for nonfringe depths values at different distances. The gaps between values are due to the discretization. (a) Depth distribution at 1 m. (b) Depth distribution at 4.5 m.

C. Measurement Noise

In this section, we use the same data collected in Section III-B to analyze how the noise term of the measured depth behaves. In particular, we will focus on four aspects: the probability distribution of the noise, its variance depending on the depth, and the magnitude of spatial and temporal correlations.

Fig. 9 shows the distributions of nonfringe depths values captured at representative positions. These are discrete distributions, and the gaps between values correspond to the discretization described in Section III-A. At short distances, the support consists of two or three values, as shown in Fig. 9(a). For higher distances, the support becomes slightly wider, reaching up to four values in Fig. 9(b). In any case, the distributions are consistently unimodal and appear roughly symmetrical around the mean (prequantization), and thus, we suspect that approximating them as Gaussians in practical applications will not lead to much loss of information.

When dealing with estimators, it is also important to know the variance that corresponds to a given measurement, preferably without having to wait for additional measurements from the same position. Fig. 10 shows the variances of the measurements gathered in the chessboard dataset from Section III-A. In yellow, we observe the variance stemming from the discretization, assuming a uniform distribution that ranges between the previous and the next possible values. In blue, we see the sample

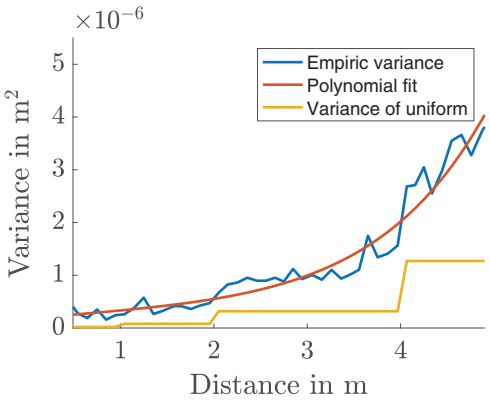


Fig. 10. Empiric variance of measurement noise at different distances, and a best-fitting polynomial fit.

variance gathered from 150 frames. In red, we see a best-fitting third degree polynomial with the form

$$\sigma_z^2 = 10^{-6} \cdot (0.07z^3 - 0.32z^2 + 0.64z - 0.02), \tag{1}$$

which closely approximates this empiric variance and can be easily integrated into an estimator.

Given a depth value z at the pixel position $[u, v]^T$, i.e., at column u and row v in the depth image, it is often necessary to obtain the covariance matrix of the reconstructed "unprojected" point $\underline{y} \in \mathbb{R}^3$ in Cartesian coordinates. This can be achieved in a closed form using the standard pinhole model [6] and the intrinsic matrix $\mathbf{K}$, which is provided directly by ARKit's API. To give the reader an idea, an example intrinsic matrix from an iPad Pro 2021 has the following form

$$\mathbf{K} = \begin{pmatrix} 212.4 & 0 & 127.0 \\ 0 & 212.4 & 96.3 \\ 0 & 0 & 1 \end{pmatrix}, \tag{2}$$

which corresponds to a horizontal field of view of approximately 60° and an image size of 256 × 192 pixels. Assuming no uncertainties, the unprojection step can be implemented by introducing a screen space vector in homogeneous coordinates

$$\underline{y}^{uv} = [u, v, 1]^T \tag{3}$$

which in turn yields

$$\underline{y} = \mathbf{K}^{-1} \cdot \underline{y}^{uv} \cdot z. \tag{4}$$

We will now extend this step to assume that $\underline{y}^{uv}$ and z are both uncertain. We assume that u and v have a distribution of $\mathcal{U}(-1, 1)$, i.e., they are uniformly distributed between the previous and following pixels. Using moment matching, we obtain

$$\mathbf{C}^{uv} = \operatorname{cov}\left(\underline{y}^{uv}\right) = \operatorname{diag}\left(\frac{1}{3}, \frac{1}{3}, 0\right). \tag{5}$$

Finally, by using the product rule of independent random variables, we propagate (1) and (5) through (4) to obtain

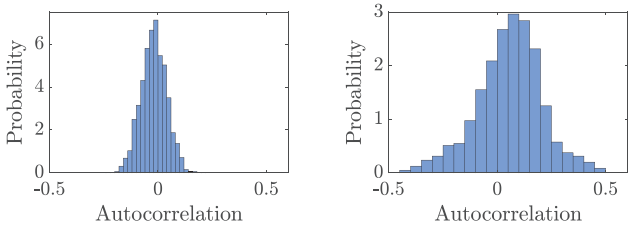
$$\mathbf{C}^{y} = \mathbf{K}^{-1}\left(\mathbf{C}^{uv}\sigma_z^2 + \underline{y}^{uv}\left(\underline{y}^{uv}\right)^T\sigma_z^2 + \mathbf{C}^{uv}z^2\right)(\mathbf{K}^{-1})^T. \tag{6}$$

D. Measurement Correlations in Time and Space

When discussing the stochastic properties of measurement noise in sensors, the topic of correlations is usually not mentioned. This omission is justified with sensors such as the Kinect 2, where measurements are mostly independent from each other. However, due to the ML interpolation system, these assumptions cannot be guaranteed to hold for the iPad. Note that, especially in probabilistic estimators, ignoring correlations can lead to estimates with misleading variances, as the estimator cannot compensate for the fact that measurements with dependent noise terms carry less information. This

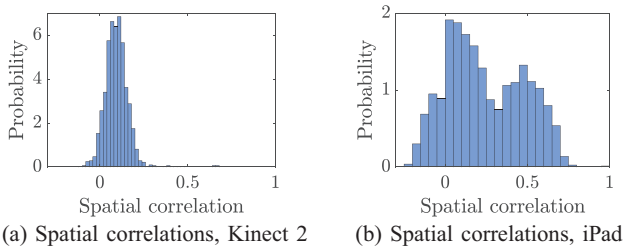
(a) Setup: Cuboid in front of a wall (b) View from above, Kinect 2 (c) View from above, iPad

Fig. 11. Setup of the cuboid in front of wall, observed by a Kinect 2 and an iPad positioned next to each other. Note how the Kinect 2 produces spatially uncorrelated noise, while the iPad produces a smooth, almost flat surface that shifts and deforms each frame. (a) Setup: cuboid in front of a wall. (b) View from above, Kinect 2. (c) View from above, iPad.



(a) Autocorrelations, Kinect 2 (b) Autocorrelations, iPad

Fig. 12. Distribution of autocorrelation coefficients for multiple pixels in a 20×20 px window around the box center and a shift of $\tau = 1$ frame. (a) Autocorrelations, Kinect 2. (b) Autocorrelations, iPad.



(a) Spatial correlations, Kinect 2 (b) Spatial correlations, iPad

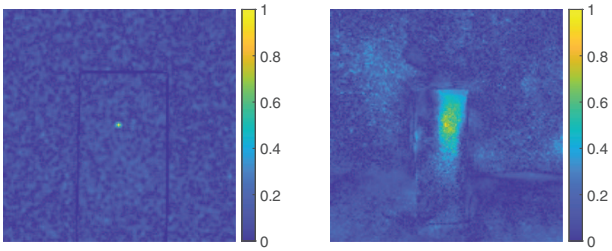
Fig. 13. Distribution of spatial correlation coefficients for multiple pixels in a 20×20 px window in relation to the center pixel. (a) Spatial correlations, Kinect 2. (b) Spatial correlations, iPad.

serves as a motivation to study these correlations more explicitly.

In order to do this, we recorded 150 frames of a rectangular cuboid standing in front of a wall with both an iPad Pro 2021 and a Kinect 2 (see Fig. 11), at a distance of about 2 m, and analyzed a small window of 20×20 px around the center the object. The Kinect 2 will serve as a baseline, as estimators using measurements from this sensor and making no assumptions about dependency have been shown to produce satisfactory reconstructions in [14], [15], and [24]. All of the considered points have a confidence of 2 (high) on the iPad depth image and lie on the cuboid's surface. As an aside, the iPad LIDAR projector was not visible from the Kinect infrared camera, which suggests that both devices do not interfere with each other.

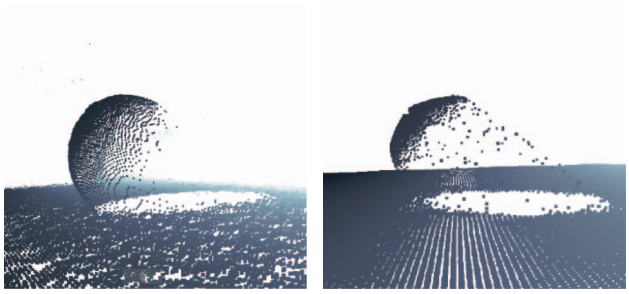
We start with autocorrelations, i.e., how much a series of measurements stemming from the same source is correlated with a time shifted version of itself. This is important given that recursive estimators with time-evolving states, such as the Kalman filter, generally assume that measurements at different time steps are independent. Fig. 12 shows how often an autocorrelation coefficient appears in a 20×20 px window around the box center, for a time shift of $\tau = 1$ frame. For the Kinect 2 [see Fig. 12(a)], we observe that for all positions the autocorrelation coefficient has an absolute value below 0.2, and most of them are below 0.1. However, for the iPad [see Fig. 12(b)], the support of the autocorrelations is wider, briefly reaching 0.5. This shows that iPad measurements (mean 0.034) are much more correlated in time than the Kinect 2 (mean -0.019). This autocorrelation fades with time, taking ten frames for the iPad autocorrelations to reach the same spread as the Kinect. Nonetheless, both are still rather close to 0 most of the time, so we consider it justifiable to assume them as time independent.

Next we will analyze how measurement samples from different pixel positions are spatially correlated. For this, we will take the correlation coefficient of all measurements in the window in relation to the measurements in the center, and tally how often a correlation coefficient appears. The results are shown in Fig. 13. The difference between both sensors, here, is much more remarkable, showing very high correlations across the board for the iPad, with the mean (0.25) being much higher than the Kinect 2 (0.09). This effect can be appreciated visually for the wall in Fig. 11. Here, for the Kinect [see Fig. 11(b), individual measurements can be seen, giving the appearance of a "dusty" cloud, while the iPad depth system generates a smooth surface that moves and deforms between frames [see Fig. 11(c)]. Furthermore, spatial correlations fade much faster for the Kinect as the distance increases. In Fig. 14(a), for example, using a larger 96×96 pixel window, we observe that all measurements farther than 3 px away from the center have a correlation with absolute value below 0.2. For the iPad, however, most of the cuboid surface retains the high correlations.



(a) Spatial correlations, Kinect 2 (b) Spatial correlations, iPad

Fig. 14. Absolute value of the spatial correlations for a 96 × 96 px window, in relation to the pixel in the center. The Kinect 2 image appears "zoomed in" because it has a higher resolution. (a) Spatial correlations, Kinect 2. (b) Spatial correlations, iPad.



(a) Sphere, Kinect 2 (b) Sphere, iPad

Fig. 15. Plastic sphere of radius 15 cm lying on the ground, being observed from a distance of 2 m by a Kinect 2 and an iPad depth sensor. (a) Sphere, Kinect 2. (b) Sphere, iPad.

If desired, spatial correlations can be incorporated into linear estimators, such as the Kalman filter and its extensions, quite easily by introducing a composite measurement and adjusting the measurement equation. Thus, if two measurements $\underline{y}_1$ and $\underline{y}_2$ are "close" to each other and known to stem from the same surface, the estimator can use

$$\underline{y} = \begin{bmatrix} \underline{y}_1 \\ \underline{y}_2 \end{bmatrix} \tag{7}$$

instead, with covariance matrix

$$\mathbf{C}^y = \begin{pmatrix} \mathbf{C}_1^y & \mathbf{C}_{1,2}^y \\ (\mathbf{C}_{1,2}^y)^T & \mathbf{C}_2^y \end{pmatrix}. \tag{8}$$

Here, $\mathbf{C}_1^y$ and $\mathbf{C}_2^y$ are calculated as before from (6). In order to derive $\mathbf{C}_{1,2}^y$, we assume that the $\underline{y}^{uv}$ components are independent from each other and from z_1 and z_2. Furthermore, it holds that

$$\operatorname{cov}(z_1, z_2) = \sigma_{z,1}\sigma_{z,2} \operatorname{corr}(z_1, z_2) \tag{9}$$

where $\operatorname{corr}(z_1, z_2)$ is the scene-dependent spatial correlation coefficient. Our experiments have shown that $\operatorname{corr}(z_1, z_2)$ usually hovers around 0.2 if the distance is less than 20 px and both measurements stem from the same surface. Finally, the correlation matrix $\mathbf{C}_{1,2}^y$ can be obtained in closed form yielding

$$\mathbf{C}_{1,2}^y = \operatorname{cov}(\underline{y}_1, \underline{y}_2) \tag{10}$$

$$= \operatorname{cov}\left(\mathbf{K}^{-1} \cdot \underline{y}_1^{uv} \cdot z_1, \mathbf{K}^{-1} \cdot \underline{y}_2^{uv} \cdot z_2\right) \tag{11}$$

$$= \left(\mathbf{K}^{-1} \cdot \underline{y}_1^{uv} \left(\underline{y}_2^{uv}\right)^T \left(\mathbf{K}^{-1}\right)^T\right) \operatorname{cov}(z_1, z_2). \tag{12}$$

E. Working with Curved Surfaces

In the previous sections, we showed that the iPad produced relatively accurate measurements from large flat surfaces, such as walls. However, due to the low spatial resolution, smaller objects (in relation to the field of view) will appear "smoothened" out, with hard corners flattened and surfaces fused together. This effect can be compensated by ensuring that the object size is much larger than 11 px, as explained in Section II-B. Still, the tendency to flatten or merge surfaces can become problematic in applications that require as much knowledge as possible about the target's shape, such as extended object tracking and classification.

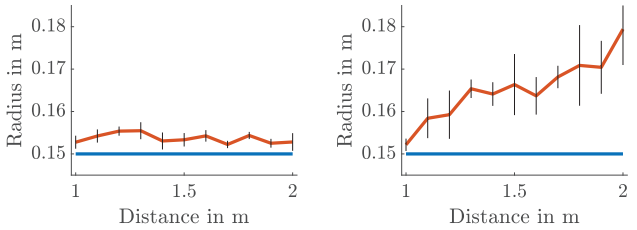
In this section, we will analyze how much information about the shape is lost at different distances by estimating the extent and position of sphere. A description of the experimental setup follows. The target sphere, with a ground-truth radius of 15 cm, was already introduced in the scene from Fig. 2. Here, instead, we place it on the floor, as shown in the example captures from Fig. 15, and observe it from a height of approximately 1.5 m. As a note, in these images, we can also appreciate the contrast between the noise correlations mentioned in Section III-D: with the Kinect 2, the floor has a highly irregular texture, whereas with the iPad, it appears almost perfectly flat.

The estimation procedure is as follows. For a given frame, the sphere is represented using the following state:

$$\underline{x} = [\underline{p}^T, r]^T \in \mathbb{R}^4, \tag{13}$$

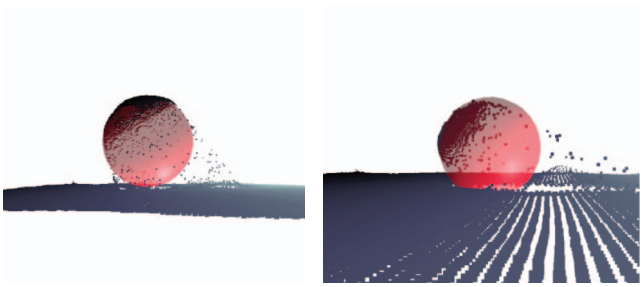
where $\underline{p}$ is the position in $\mathbb{R}^3$ and r is the radius. The following preprocessing steps are executed. First, the screen measurements are unprojected into $\mathbb{R}^3$ using (4) and (6). Second, the ground plane is estimated using RANSAC, and all pixels that belong to it with a threshold of 2 cm are removed. Third, we also eliminate fringe measurements (flying pixels), defined here as any pixel with at least one neighbor farther than 2 cm away. Finally, we use the remaining n measurements $\underline{y}_i$ for $1 \leq i \leq n$ to estimate the state $\underline{x}$ using least squares shape fitting. Here, the idea is to minimize the weighted sum of the squared residuals

$$R_i^2 = \left\| \underline{y}_i - \underline{p} \right\|^2 - r^2, \tag{14}$$



(a) Estimated radius, Kinect 2 (b) Estimated radius, iPad

Fig. 16. Estimated radius of the sphere for the Kinect 2 and the iPad, viewed from different camera distances. Ground truth in blue, result means in red, and standard deviations in black (vertical lines). (a) Estimated radius, Kinect 2. (b) Estimated radius, iPad.



(a) Sphere at 1 m (b) Sphere at 2 m

Fig. 17. Point cloud of the plastic sphere being observed by the iPad depth sensor. Best-fitting sphere overlaid in red. Floor appears wider at 2 m due to the field of view. (a) Sphere at 1 m. (b) Sphere at 2 m.

where $||\cdot||^2$ is the square of the Euclidean norm, using the inverse of the residual variances as weights

$$w_i = \frac{1}{\mathrm{var}(R_i^2)} = \frac{1}{\mathrm{tr}(\mathbf{C}_i^y)}. \tag{15}$$

The resulting $\underline{x}$ can be obtained with standard nonlinear minimization. This state estimation procedure is repeated along multiple frames as the camera moves horizontally from a distance of 1 to 2 m away from the sphere. For the sake of simplicity, the measurements are assumed to be independent from each other.

Fig. 16 shows the results at distances in intervals of 10 cm. The ground truth of 15 cm in blue, the means are shown in dark red, and the standard deviations in vertical red lines. Note that, due to the presence of artifacts, the size of the sphere can vary moderately even in consecutive frames. Furthermore, by eliminating fringe pixels, we remove measurements around the sphere border. Thus, it would be expected for the estimated radius to be lower than the ground truth, which stands in contrast with both results. This effect, however, is compensated in the opposite direction by the extent bias caused by measurement noise, a known effect in shape fitting studied in [9] and [25] among others. This bias appears constant for the Kinect 2 [see Fig. 16(a)], where the radius is consistently around 5 mm (3%) higher than the ground truth. However, the iPad, while producing accurate results at around 1 m, quickly starts losing accuracy [see Fig. 16(b)] as the camera moves away. This is a consequence of the sensor flattening the measured surface as it merges into the floor, an effect also observed in previous experiments, which in turn increases the size of the estimated sphere.

Example results can be seen in Fig. 17, with the estimated sphere in red. Here, we can see how, at a short distance, measurement quality is comparable to the Kinect in Fig. 15(a), yielding a radius estimate of 15.5 cm. However, after a short distance, measurements become more sparse and the proportion of fringe pixels increases significantly. Furthermore, the patch becomes so flattened that the increased radius (17.5 cm) pushes the sphere position into the ground. Note that, at this distance, the sphere is only 32 px wide. After this point, the radius reaches 20 cm at a distance of 2.3 m, and beyond that segmentation starts to become difficult, given much of the sphere has merged into the floor.

IV. CONCLUSION

In this paper, we presented a quantitative analysis of the spatial uncertainties for the iPad Pro depth sensor. As motivation, we explained how Apple, with its high market share in mobile devices, has begun shipping depth sensors integrated in their smartphones and tablets, increasing the reach of applications in localization, tracking, and classification without extra cost to developers and users. Thus, it makes sense to analyze the properties, benefits, and pitfalls of these new data streams. First, we briefly described the direct ToF sensor that the tablet uses to obtain depth images, and pointed out that the depth stream is most likely extrapolated from only 576 real measurements, which would explain the observed low spatial resolution. We also noted that the values provided by the API are not the direct measured values, and instead, they are generated by an ML algorithm that incorporates information from the color stream. As part of the analysis, we measured the discretization of the depth domain, provided a model for the measurement bias and error variance, and described the temporal and spatial correlations between measurements. We also showed the tendency of the iPad depth sensor to merge and flatten surfaces, which is useful when dealing with planes such as walls, but becomes problematic when estimating the shape of curved objects, such as spheres.

In general, it can be seen that the iPad depth sensor, in its current iteration, is well suited for simultaneous localization and mapping (SLAM) based on planar surfaces. This is shown by the robust camera tracking provided by the default libraries. Dealing with other objects, however, imposes some restrictions on their size and how far they can be from the sensor. Curved objects, or objects with steep depth changes, appear flattened out, which can reduce its applicability in fields, such as extended object tracking or object classification based on point clouds, which have higher requirements on measurement quality. Thus, applications that deal with these

objects should keep in mind the tightened operating range. Still, these limitations should be balanced with the advantages provided by the Apple ecosystem, and the wide reach of potential users available to applications using it.

REFERENCES

[1] 4da.tech, "LiDAR: Apple LiDAR Analysis," 2021. Accessed: Oct. 26, 2021. [Online]. Available: https://3da.medium.com/lidar-apple-lidar-and-dtof-analysis-cc18056ec41a.

[2] Apple, "iPad Pro 2020," Accessed: Oct. 26, 2021. [Online]. Available: https://www.apple.com/de/ipad-pro/

[3] Apple, "ARKit," 2021. Accessed: Oct. 26, 2021. [Online]. Available: https://developer.apple.com/augmented-reality/

[4] A. Breitbarth, T. Schardt, C. Kind, J. Brinkmann, P.-G. Dittrich, and G. Notni "Measurement accuracy and dependence on external influences of the iPhone X TrueDepth sensor," in Proc. Photonics Educ. Meas. Sci., 2019, pp. 1114407.

[5] A. Corti, S. Giancola, G. Mainetti, and R. Sala "A metrological characterization of the Kinect v2 time-of-flight camera," *Robot. Auton. Syst.*, vol. 75, pp. 584–594, 2016.

[6] K. M. Dawson-Howe and D. Vernon "Simple pinhole camera calibration," *Int. J. Imag. Syst. Technol.*, vol. 5, no. 1, pp. 1–6, 1994.

[7] D. Eigen, C. Puhrsch, and R. Fergus "*Depth map prediction from a single image using a multi-scale deep network*," in Proc. Proc. 27th Int. Conf. Neural Inf. Process. Syst., 2014, pp. 2366–2374.

[8] F. Faion, S. Friedberger, A. Zea, and U. D. Hanebeck "Intelligent sensor-scheduling for multi-Kinect-tracking," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2012, pp. 3993–3999.

[9] F. Faion, A. Zea, and U. D. Hanebeck "Reducing bias in Bayesian shape estimation," in *Proc. 17th Int. Conf. Inf. Fusion (FUSION)*, 2014, pp. 1–8.

[10] R. Garg, V. K. Bg, G. Carneiro, and I. Reid "Unsupervised CNN for single view depth estimation: Geometry to the rescue," in Proc. Eur. Conf. Comput. Vis., 2016, pp. 740–756.

[11] C. Godard, O. Mac Aodha, and G. J. Brostow "Unsupervised monocular depth estimation with left-right consistency," in *Proc. IEEE Conf. Comput. Vis. pattern Recognit.*, 2017, pp. 270–279.

[12] C. Gollob, T. Ritter, R. Krabnitzer, A. Tockner, and A. Nothdurft "Measurement of forest inventory parameters with Apple iPad pro and integrated LiDAR technology," *Remote Sens.*, vol. 13, no. 16, pp. 3129, 2021.

[13] Google, "ARCore," 2021. Accessed: Oct. 26, 2021. [Online]. Available: https://developers.google.com/ar

[14] S. Izadi et al., "Kinectfusion: Real-time 3D reconstruction and interaction using a moving depth camera," in *Proc. 24th Annu. ACM Symp. User Interface Softw. Technol.*, 2011, pp. 559–568. [Online].

[15] S.-Y. Jiang, N. Y.-C. Chang, C.-C. Wu, C.-H. Wu, and K.-T. Song "Error analysis and experiments of 3D reconstruction using a RGB-D sensor," in *Proc. IEEE Int. Conf. Automat. Sci. Eng.*, 2014, pp. 1020–1025.

[16] N. Kitsunezaki, E. Adachi, T. Masuda, and J. Mizusawa "Kinect applications for the physical rehabilitation," in *Proc. IEEE Int. Symp. Med. Meas. Appl.*, 2013, pp. 294–299.

[17] G. Luetzenburg, A. Kroon, and A. A. Bjork "Evaluation of the Apple iPhone 12 pro LiDAR for an application in geosciences," *Sci. Rep.*, vol. 11, no. 1, pp. 1–9, 2021.

[18] L. Gregor, K. Aart, and B. Anders, "Lidar scanning of coastal cliffs in Denmark and Greenland with the Apple iPhone 12 pro," in *Proc. AGU Fall Meeting 2021*, 2021.

[19] R. Lun and W. Zhao "A survey of applications and human motion recognition with Microsoft Kinect," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 29, no. 05, 2015, Art. no. 1555008.

[20] E. Marder-Eppstein "Project tango," in *Proc. ACM SIGGRAPH 2016 Real-Time Live!*, 2016, Art. no. 40.

[21] A. Matsuura, M. Dan, A. Hirano, Y. Kiku, S. Torii, and S. Morita "Body measurement of riding horses with a versatile tablet-type 3D scanning device," *J. Equine Sci.*, vol. 32, no. 3, pp. 73–80, 2021.

[22] I. Moazzam, K. Kamal, S. Mathavan, S. Usman, and M. Rahman "Metrology and visualization of potholes using the Microsoft Kinect sensor," in *Proc. 16th Int. IEEE Conf. Intell. Transp. Syst.*, 2013, pp. 1284–1291.

[23] A. Murtiyoso, P. Grussenmeyer, T. Landes, and H. Macher "First assessments into the use of commercial-grade solid state LIDAR for low cost heritage documentation," *ISPRS-Int. Archives Photogrammetry Remote Sens. Spatial Inf. Sci.*, vol. 43, pp. 599–604, 2021.

[24] C. V. Nguyen, S. Izadi, and D. Lovell "Modeling Kinect sensor noise for improved 3D reconstruction and tracking," in *Proc. Second Int. Conf. 3D Imaging, Model., Process., Visual. Transmiss.*, 2012, pp. 524–530.

[25] T. Okatani and K. Deguchi "On bias correction for geometric parameter estimation in computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2009, pp. 959–966.

[26] J.-H. Park, Y.-D. Shin, J.-H. Bae, and M.-H. Baeg "Spatial uncertainty model for visual features using a Kinect sensor," *Sensors*, vol. 12, no. 7, pp. 8640–8662, 2012.

[27] A. Spreafico, F. Chiabrando, L. Teppati Lose, and F. Giulio Tonolo "The iPad pro built-in LIDAR sensor: 3D rapid mapping tests and quality assessment," *ISPRS-Int. Archives Photogrammetry Remote Sens. Spatial Inf. Sci.*, vol. 43, pp. 63–69, 2021.

[28] *System Plus Consulting*, "Apple iPad Pro LiDAR module," 2021. Accessed: Oct. 26, 2021. [Online]. Available: https://www.systemplus.fr/reverse-costing-reports/apple-ipad-pro-11s-lidar-module/

[29] S. Tatsumi, K. Yamaguchi, and N. Furuya "ForestScanner: A mobile application for measuring and mapping trees with LiDAR-equipped iPhone and iPad," *Methods in Ecology and Evolution*, 2021. https://besjournals.onlinelibrary.wiley.com/doi/abs/10.1111/2041-210X.13900

[30] M. Vogt, A. Rips, and C. Emmelmann "Comparison of iPad Pros LiDAR and TrueDepth capabilities with an industrial 3D scanning solution," Technologies, vol. 9, no. 2, 2021. Art. no. 25. [Online]. Available: https://www.mdpi.com/2227-7080/9/2/25

[31] X. Wang et al., "Evaluation of ipad pro 2020 LIDAR for estimating tree diameters in urban forest," , *ISPRS Ann. Photogrammetry Remote Sens. Spatial Inf. Sci.*, vol. 8, pp. 105–110, 2021.
[32] O. Wasenmuller and D. Stricker "Comparison of Kinect v1 and v2 depth images in terms of accuracy and precision," in *Proc. Asian Conf. Comput. Vis.*, 2016, pp. 34–45.
[33] X. Xu, A. Al-Dahle, and K. Garg "Shared sensor data across sensor processing pipelines," U. S. Patent 10 671 068, 6 2, 2020.

Dr.-Ing. Antonio Zea is a researcher at the Intelligent Sensor-Actuator-Systems (ISAS) Laboratory of the Karlsruhe Institute of Technology (KIT), where he also obtained his Ph.D. degree in the field of Computer Science in 2018. His research interests include extended object tracking, localization, robot teleoperation with virtual and augmented Reality, and human machine interaction.

Uwe D. Hanebeck is currently a Chaired Professor of computer science with the Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, and the Director of the Intelligent Sensor-Actuator-Systems Laboratory (ISAS), Karlsruhe, Germany. His research interests include information fusion, nonlinear state estimation, stochastic modeling, system identification, and control with a strong emphasis on theory-driven approaches based on stochastic system theory and uncertainty models.