Dynamic Real-time Spatio-Temporal Acquisition and Rendering in Adverse Environments*

Somnath Dutta[1 [0000−0003−3982−8780]], Fabio Ganovelli[1 [0000−0002−0378−9188]], and Paolo Cignoni[1 [0000−0002−2686−8567]]

Institute of Information Science and Technologies, "Alessandro Faedo" (ISTI), Italian National Research Council (CNR),Via Giuseppe Moruzzi 1, 56124 Pisa, Italy
{firstname.lastname}@isti.cnr.it

Abstract. This paper introduces NausicaaVR, a novel hardware/software system designed to acquire and render intricate 3D environments, with a particular emphasis on challenging and adverse contexts. In doing so, we navigate the complex landscape of system calibration and rendering, while seamlessly integrating data from multiple sensors. We explore the distinctive challenges inherent in adverse environments, juxtaposing them against conventional automotive scenarios. Through a comprehensive exposition of all constituent elements of the NausicaaVR system, we offer transparent insights into the encountered obstacles and the intricate decisions that were instrumental in surmounting them. This study seeks to illuminate the developmental trajectory of NausicaaVR and analogous systems, thereby furnishing a repository of knowledge and understanding poised to benefit future research and the pragmatic implementation of such cutting-edge technologies.

Keywords: Multi-Sensor Calibration · Real-time Rendering · Virtual Reality.

1 INTRODUCTION

In a multi-sensor environment, perception and rendering play crucial roles in understanding and representing the surrounding world. With the advancements in sensor technologies, such as cameras, lidar, radar, and depth sensors, the ability to capture rich and diverse data about the environment has greatly expanded. Multi-sensor perception involves the integration and fusion of data from multiple sensors to generate a comprehensive understanding of the scene, including the detection and tracking of objects, estimation of their poses and velocities, and the creation of detailed 3D models. On the other hand, rendering in a multi-sensor environment aims to create realistic and immersive visual representations of the perceived scene, taking into account the different sensor modalities and their respective characteristics. This involves techniques such as sensor fusion,

* NAUSICAA - NAUtical Safety by means of Integrated ComputerAssisted Appliances 4.0 (DIT.AD004.136)

data alignment, mapping, and rendering algorithms to generate accurate and visually appealing virtual representations of the environment. The combination of multi-sensor perception and rendering enables applications in various domains, including autonomous driving [6], virtual reality, augmented reality, and robotics, where an accurate understanding and realistic visualization of the environment are paramount.

Multi-sensor perception and rendering pose several challenges due to the complexity of integrating data from multiple sensors and synthesizing a coherent representation of the environment. Some of the key Challenges include:

- **Data Fusion**: Combining data from different sensors with varying characteristics. The challenge lies in aligning and synchronizing the data streams, handling differences in resolution, accuracy, and noise levels, and resolving conflicts or inconsistencies between sensor measurements.
- **Sensor Calibration**: Accurate calibration of sensors is essential for achieving reliable multi-sensor perception. Ensuring that the sensors are properly aligned, calibrated, and synchronized is a non-trivial task. Sensor calibration involves estimating intrinsic and extrinsic parameters, such as camera intrinsics, lidar calibration, and sensor-to-sensor transformations.
- **Occlusion and Sensor Limitations**: Dealing with occlusions and handling sensor limitations are important challenges in multi-sensor perception. Occlusions can lead to missing or incomplete data, requiring techniques to infer or reconstruct occluded regions. Moreover, sensor limitations, such as limited field of view, range, or resolution, need to be considered to ensure accurate perception and rendering of the environment.
- **Real-time Performance**: Multi-sensor perception and rendering systems often operate in real-time applications such as robotics, autonomous vehicles, or augmented reality. Achieving real-time performance while maintaining accuracy and reliability is a challenge. Efficient algorithms, optimization strategies, and hardware acceleration are necessary to meet the stringent timing requirements.

With the above-mentioned objective in focus, we highlight the substantial outcomes of our work as summarized below.

- a testbed architecture with multiple cameras and low-cost lidar sensors, described in section 3.1
- an ad hoc method (Section 4) for registration of input data in a common reference frame

The paper is organized into multiple sections focusing on related works in section 2, a description of the overall system-cum-architecture in section 3 followed by calibration of lidars and cameras in section 4.

2 RELATED WORK

In this section, we aim to provide a comprehensive overview of existing literature, primarily focusing on two key aspects: the system framework and the

spatio-temporal calibration of a multi-modal sensor system. Regarding the system framework, we will explore previous works that have proposed various architectures, designs, or methodologies for integrating multiple sensors within a cohesive system. This includes studies that have investigated the fusion of data from different modalities, the synchronization of sensor outputs, or the development of algorithms for real-time processing and analysis.
Additionally, we will delve into the topic of spatio-temporal calibration in multi-modal sensor systems. This area of research deals with the alignment and synchronization of spatial and temporal data captured by different sensors. We will examine different calibration techniques and algorithms proposed in the literature, along with their advantages, limitations, and potential applications.
System-Framework Visual sensors are essential components in maneuvering operations across diverse scenarios, such as street navigation with automotive vehicles [19], aerial navigation with drones [26], and marine operations in boats and ships [17]. In recent years, rapid advancements in technologies such as sensing devices, Artificial Intelligence (AI), and the Internet of Things (IoT) have sparked significant transformations across various domains, leading to their widespread adoption in diverse applications. Tonnis et al. [41] emphasized the increasing significance of spatial sensor systems in cars, forming the basis for safety and driver assistance systems. The authors present their developed visualization system for spatial sensor data, incorporating various setups and visualization devices to ensure precise spatial alignment and support the advancement of driver assistance systems. Vu et al. [42] presented a multi-sensor-based approach for object perception in automotive applications, addressing the detection, tracking, and classification of objects while considering various classes. The proposed method employs fusion techniques to combine information from lidar and camera sensors, resulting in a more reliable representation of detected objects in real-life scenarios.
Our literature research also revolves around marine maneuver and navigation, aligning with our research project on a similar topic. However, our framework takes a more generic approach, designed to accommodate variability with necessary modifications. In the field of marine vessels, there has been active research and development of technology-related autonomous ships ([36],[13]) to enhance safety by mitigating human errors and improving working conditions through reduced crew workload. One noteworthy application-oriented research and development project is SmartKai [8], which is centered around creating a parking system for ships at the harbor, employing lidar sensors. The project includes the development of a smart user interface, enabling ship crews to effortlessly visualize navigation data across various display platforms. Moreover, in a study by [37], a camera-based visual sensing system is proposed to cater to maritime navigation and reconnaissance applications, including obstacle avoidance and area survey analysis.
Ruessmeier et al. [34] conceptualized and implemented an experimental maritime testbed for sensor data fusion, communication technology, and data stream analysis tools. The setup is highly flexible and applicable in various research fields,

including e-navigation and situational awareness generation. Brinkmann et al. [3] introduced LABSKAUS, a maritime physical testbed/cyber-physical system, offering maritime-specific components like a reference waterway, research boat, and mobile bridge. The proposed architecture includes a data model, message parser, wireless infrastructure, and a polymorphic interface, enabling the integration of various prototype designs within LABSKAUS. [21] discuss digitalization in marine vessels as a significant process directed toward autonomous navigation, cost reduction, safety, and reliability. The authors point out that the complete system consisting of advanced sensors, Artificial Intelligence, and alternative display techniques (VR, AR) is a major requirement in the marine intelligence system, but also pitches an enormous challenge for integration and deployment. [29] proposed a simple hardware system and software architecture for collecting the sensor data (non-visual) targeting autonomous surface vessels (ASVs). Furthermore, a human-machine interface (HMI) is implemented as part of the system.

A chronological trend of the ASV's existing autonomy levels in marine vessels and multi-agent control architecture from the perspective of ASVs is presented in [35]. According to the authors, situation awareness that forms an integral block of the navigation systems heavily relies on sensor fusion and the corresponding data visualization. [38] presents a detailed review of the sensor and AI technique for environment perception and awareness for autonomous ships. [43] explores the use of AI techniques to integrate multiple sensor modalities into a cohesive approach for autonomous ship navigation. The use of multiple redundant sensors overcomes the limitations and vulnerabilities of the individual sensor and the usage of advanced learning methodology addresses key areas of detection and identification providing comprehensive situational awareness to be effective in real-time maneuvering.

Multi-Sensor Calibration: In order to effectively integrate information (spatially and temporally) obtained from multiple sensing modalities, it is essential to represent them in a common reference frame. The problem of estimation of the rigid body transformation between the multimodal sensory information (camera and LiDAR) has been extensively studied in the past two decades [16], [22], [30]. Despite recent developments, fusing multimodal sensory information is still a challenging problem [33]. [23] proposed a framework tailored for global-shutter camera and 3D LiDAR setups with fixed internal camera calibration parameters and an unknown but constant time offset between the sensors. Kodaira et al.[18] proposed a segmentation-based framework to jointly estimate geometric and temporal parameters for calibrating a camera-LIDAR sensor suite, achieving accurate real-time calibration without the need for calibration labels. The primary limitation lies in its dependency on high-quality semantic segmentation masks, which may impact calibration accuracy, particularly in scenarios with compromised segmentation performance. Grammatikopoulos et al. [12] presented a straightforward method to calibrate Lidar-camera systems using AprilTag markers and a custom retro-reflective target. The approach achieves geometric alignment and temporal synchronization, demonstrated on a four-

camera mobile mapping system with integrated Velodyne Lidar for accurate multi-camera point cloud texturing. A trihedral object's geometric constraint is employed to achieve a calibration through nonlinear square optimization of a 3D lidar-camera system in [11]. While the method does rely on minimal manual input, it's important to note that in scenarios involving irregular or complex environments, the trihedral assumption may not remain valid. In such cases, the method's performance could decline, particularly if the initial plane region inputs are significantly inaccurate. The research from [27] addresses the limitations of traditional calibration methods by introducing a novel targetless, structureless approach for spatio-temporal alignment between LiDAR and visible cameras on robotic systems. Unlike methods assuming scene geometry, this approach accommodates various sensor configurations and environmental conditions, showcasing accuracy in estimating spatio-temporal parameters.

This paper extends the work presented in [9] introduces a comprehensive framework centered around affordable sensors. In contrast to existing methodologies, our approach inherently tackles the challenge of registering and calibrating multi-modal data. This is particularly crucial given the issues of low resolution, lidar data sparsity, and complex environmental conditions. We provide detailed insights into the alignment of multi-modal data in Section 4, carefully considering the aforementioned challenges. This is achieved through a combination of tailored calibration objects and a streamlined algorithmic approach. Importantly, we propose a calibration refinement procedure based on the photo-consistency model as elaborated in section 4.3 that reduces the overall calibration errors and ghosting artifacts induced by asynchronous data collection.

3 NAUSICAAVR-FRAMEWORK

The schematic diagram 1 offers a holistic view of our entire framework. Subsequent sections provide in-depth elucidation of our framework, encompassing the hardware configuration in section 3.1 and the proprietary software interface section 3.2.

Our hardware configuration entails a multi-modal sensor system integrating two Lidar scanners and four embedded color cameras, each equipped with fish-eye lenses. To be precise, we employ Velodyne's VLP-16 PUCK LITE lidar scanners and Imaging Source cameras [14], which are purpose-built for operating effectively in demanding environmental conditions.

3.1 Sensor System Configuration

The cameras as shown in Figure 1 are interfaced with NVIDIA Jetson embedded hardware [25], running on the Linux Tegra OS. NVIDIA's Jetson hardware and its extensive software development kits (SDKs) also cater to Artificial Intelligence (AI) applications, rendering them exceptionally well-suited for autonomous machines and integrated systems. The captured video signals from

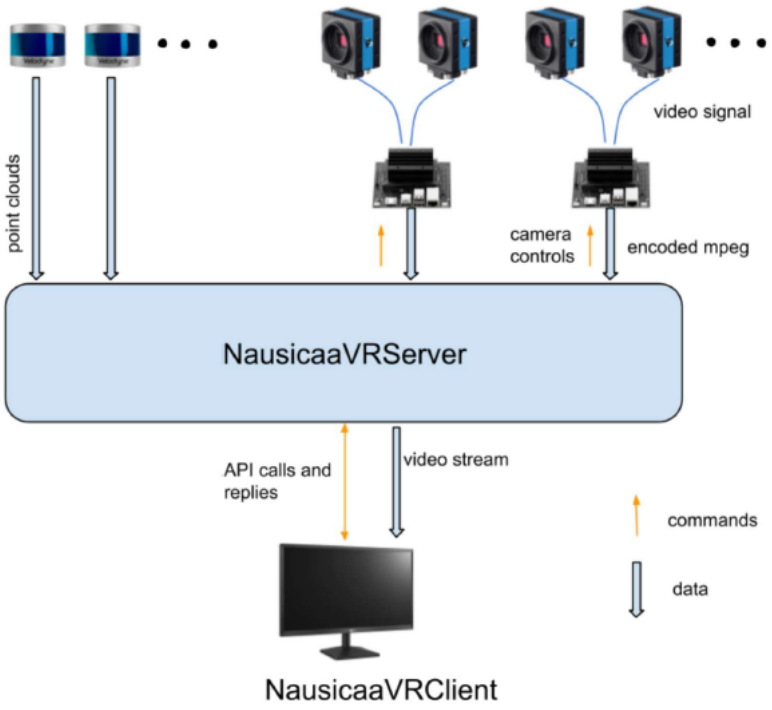

Fig. 1: Hardware-Software System (from [9])

the cameras undergo H264 encoding, facilitating efficient transmission of the resulting streams over a wired network to the server.

With the hardware acceleration capabilities of Jetson, the video stream is rapidly encoded, enabling seamless transmission of HD-resolution data at 60fps. The parameters of each individual camera sensor are optimally refined through a manual process, leveraging information from the camera software development kit (SDK) and tailored to suit the specific acquisition environment. Moreover, the H264 encoding and streaming parameters are fine-tuned to capitalize on the hardware encoding acceleration capabilities offered by Nvidia Jetson. These carefully calibrated camera sensor data and encoding parameters are seamlessly integrated using the GStreamer open-source multimedia framework pipeline, enabling efficient data transmission as UDP packets.

Concurrently, the individual lidars (VLP-16) proficiently stream real-time 3D point data as UDP packets via ethernet. Subsequently, an Intel core-i9 PC, complemented with a powerful Nvidia GeForce RTX 3090 graphics card (24GB) and running on the Windows 11 Platform, is designated to receive UDP packets from each sensor for further processing and visualization. This setup enables the system to execute data processing tasks with utmost precision and present a comprehensive visualization of the acquired data.

3.2 Software Configuration

The data that streams from both the lidars and cameras undergoes further processing and visualization through our proprietary application framework. This framework serves as a robust platform primarily dedicated to the real-time rendering of lidar point clouds and the visualization of camera streams. Importantly, it facilitates alignment operations for both lidar-lidar and camera-lidar data, a crucial step that enables us to effectively map the geometric data extracted from the 3D point cloud and the visual texture information derived from the camera images. This intricate process culminates in the creation of a realistic rendered view of the environment. Moreover, our framework extends its capabilities to encompass remote visualization, achieved through MJPEG streaming accessible via standard web browsers. This feature enables users to remotely view the rendered data without the need for specialized software. Crucially, the framework establishes a seamless communication channel between client applications and the Server PC. This interaction is made possible through the utilization of network socket connections and advanced API functionalities. This sophisticated communication mechanism empowers clients to actively engage with the application hosted on the server PC. It allows clients to virtually explore the scene from various camera viewpoints, providing them with a dynamic and interactive experience. Furthermore, the system facilitates the reception of essential feedback, enhancing the interactivity and usefulness of the application.

4 SPATIO-TEMPORAL REGISTRATION OF LIDARS AND RGB CAMERAS.

The calibration challenge involving Velodyne VLP16 is widely acknowledged in the literature, as evident from studies like [1, 28, 20]. This challenge primarily revolves around establishing 3D-2D correspondences between the LIDAR-generated point cloud and RGB camera images. The complexity of this task is influenced by several factors:

- **LIDAR resolution:** Determining the 3D position of a target point from a point cloud relies on inferred information. Sparse point clouds pose more difficulty in locating target points accurately. The more the latter is sparse, the more difficult it is to find the target point.
- **Sensor disposition:** Since the target points need to be seen both from the LIDAR and the camera, their relative position influences the sparsity of the point cloud. As an example, in practical scenarios where the camera is typically situated at a distance from the LIDAR and faces away from it, the sampling on the target will be less dense.
- **Environment conditions:** In a controlled environment where LIDARs and cameras are present, we have the flexibility to create customized configurations that simplify the process of finding correspondences. As an illustration, within an empty room, we could utilize the corner points located at the juncture of walls and either the floor or ceiling.

The challenges in our particular scenario include all the aforementioned factors. The VLP16's limited vertical axis resolution and depth precision contribute to the intricacy. Additionally, the spatial separation between cameras and LIDARs compounds the issue, often placing them at considerable distances. Furthermore, the need for calibration in adverse environments, such as on a boat over water, exacerbates the difficulties due to the lack of reference points. Given the intricacies posed by these factors, our strategy entailed the development of a custom-designed target. This target strikes a balance between portability and visibility at a distance, as elaborated in the following section.

4.1 Calibration Target Design

We aimed to create a tangible target that could be consistently and automatically detected in both point clouds and images. Simultaneously, it needed to be simple to construct and lightweight to be usable with commercial drones. The initial prototype took the form of a cusp, formed by the intersection of three non-coplanar planar cardboard pieces (depicted in Figure 2, left). A similar approach was proposed in [4]. This design choice was based on the principle that even incomplete sampling of the three planar regions would establish their supporting planes and consequently the cusp point (the intersection of these planes). However, although this method functioned to some extent, we observed that the accuracy of the detected cusp point was compromised even at relatively short distances. This inaccuracy stemmed from the precision of LIDAR values, whereby fitting planes to a relatively small spatial region (approximately 100 points) could lead to multiple equally valid fitting planes due to the granularity of measurements. As a result, the intersection of these planes defined a point with a radius spanning several centimeters even when positioned just 3 meters from the LIDAR.

Fig. 2: Left: preliminary version of the target; Right: refined target used with our system (from [9]).

Our designed target strategically leverages the LIDAR's most precise scan directions, particularly the azimuthal direction, while remaining robust against

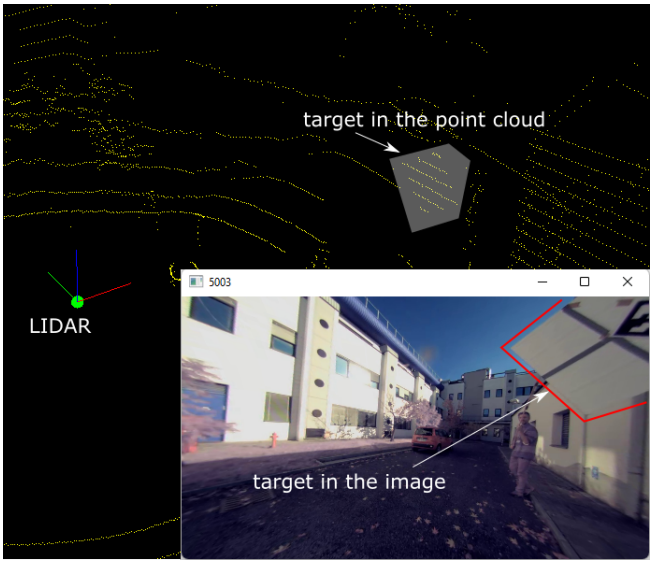

Fig. 3: Example of how the target appears in the point cloud and in one of the RGB images (from [9]).

issues like vertical low-frequency sampling and potential depth measurement inaccuracies. The approach involves employing a straightforward $1m \times 1m$ square suspended from one corner, with its center serving as the target point (as illustrated on the right side of Figure 2). An example of the resulting point cloud is shown in Figure 3. The subsequent procedure involves the following steps:

1. **Fitting a Plane:** A plane is fitted using the sampling points of the square.
2. **Point Projection:** The points are projected onto the fitted plane.
3. **Point Cloud Rotation:** The projected point cloud on the plane is rotated to minimize the size of the 2D bounding box.
4. **Target Identification:** If the size of the bounding box falls within a predetermined tolerance from the expected dimensions of $1m \times 1m$, the center of the bounding box is designated as the target point.

The described algorithm essentially performs a fitting of a fixed-size square on the point cloud, using the bounding box as a cost function. Note that, in its simplicity, this method has two useful qualities. It only needs a partial sampling of the points along the sides in order to be detected. It is tolerant of errors in in-depth measurement. This latter statement can be supported by sketching a simple proof. Let us consider the scheme in Figure 5 showing the 2-dimensional case. The actual supporting line (that is, the actual plane of the target) is $\mathbf{L}$ but because of the imprecise depth values, we fit the point with line $\mathbf{L}_f$. Let h be the thickness of the slab of points that are supposed to be on the same line. We can find the angle between the worst fitting line $\mathbf{L}_F$ and $\mathbf{L}$ as $\alpha_{err} = arctan(h/0.5)$. It follows that the projection of the point on this line will be erroneously scaled

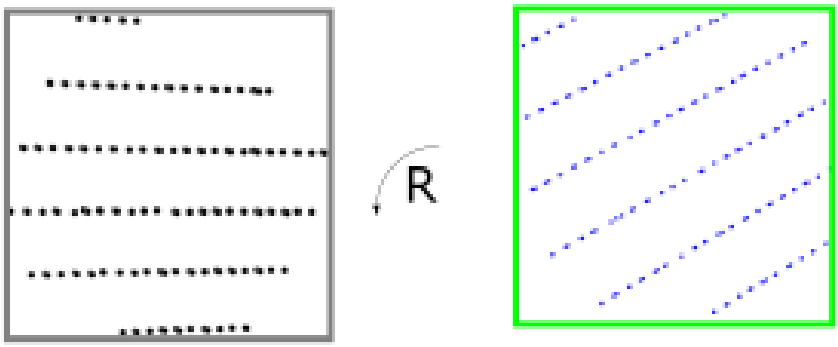

Fig. 4: Left: Points on the target projected on their fitting plane. Right: Point rotated to fit the (known) bounding box of the target (from [9]).

by $cos(\alpha_e rr)$, that is $\|p_f\| = \|p\| * cos(\alpha_{err})$. Now, to put things in perspective, consider that at 4 meters from the LIDAR, we can have a depth error around $0.03m$, which gives an error of $cos(arctan(0.03/0.5)) = 0.998$, which means that we can have the square shrunk at most by $2mm$.

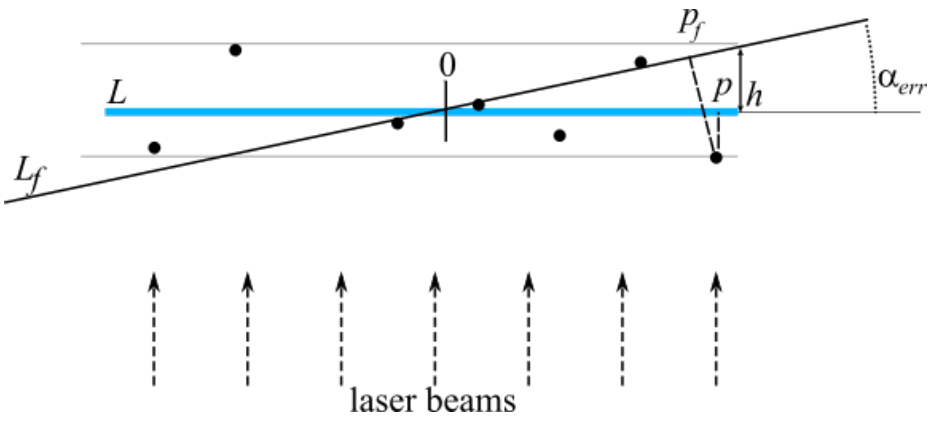

Fig. 5: Proof sketch that approximated depth measurement has limited effect on the computation of the target size (from [9]).

Detecting the same target in the RGB images is an easier problem which is solved with consolidated markers such as the Aruco markers [10].

4.2 Calibration procedure

The calibration process involves showing the target to the LIDARs and cameras until an adequate number of correspondences are accumulated for data alignment. During initialization, the user is prompted to indicate the point cloud region containing the target through a straightforward mouse click. Subsequently, continuous tracking of the target within the point clouds is established.

Whenever the target is identified in both point clouds, a fresh 3D-3D correspondence is gathered and used for point cloud alignment. Should the target be located in at least one point cloud, its corresponding 2D point in the images is sought. This leads to the creation of a 3D-2D correspondence for each image where the target point is located.

In theory, just four 3D-3D correspondences are required for point cloud alignment, and the same number of 3D-2D correspondences are needed for each image. However, the alignment's efficacy is heavily reliant on the precision and distribution of these correspondences. For instance, nearly quasi-collinear 3D points can yield unstable point cloud alignment, while 2D points concentrated within a small image area may result in inaccuracies. As such, the aforementioned simple algorithm necessitates further elaboration to accommodate these intricacies.

Acquisition time. Given that our data is collected from various sources asyn-

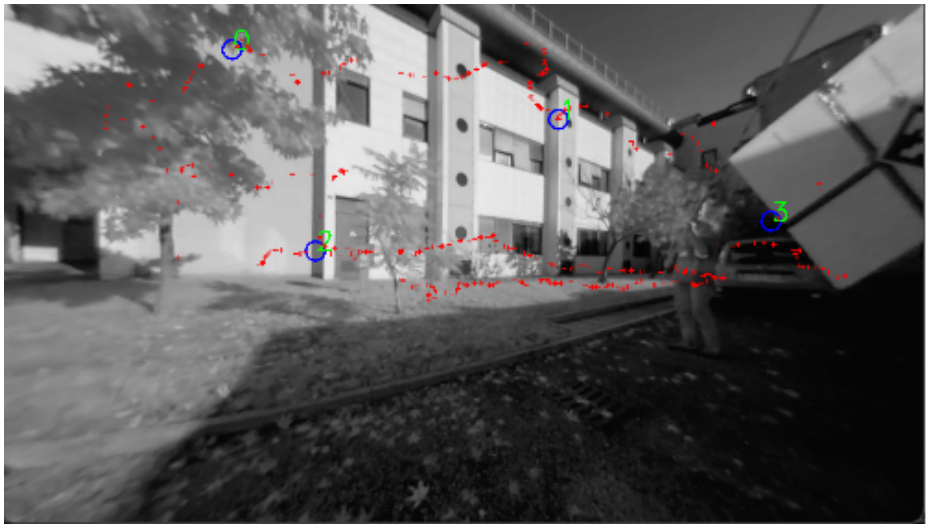

Fig. 6: All the correspondences found between the image and the 3D geometry as red dots. Selected correspondences are rendered with blue circles. The greyscale version of the image is rendered in order to better highlight the correspondence points (from [9]).

chronously, ensuring their simultaneous acquisition is not feasible, and often not

the case. This becomes particularly problematic when the target is in motion, as it can lead to erroneous correspondences.

To address this challenge, we determine the target's speed and utilize its position only when it exhibits extremely slow movement (e.g., 5 cm/s in our tests). Furthermore, we evaluate the timestamps of point clouds and associated data, discarding correspondences if the time interval between the point cloud and the corresponding images surpasses a predefined threshold (e.g., 100 ms in our experiments). These strategies are implemented to mitigate the impact of asynchronous data acquisition from various sources.

Distribution of correspondences. For both 3D-3D and 3D-2D alignments, it is imperative to possess a sparse set of correspondences. Additionally, these correspondences should avoid configurations that could lead to degeneracy, such as collinear points (for both types of alignments) or 3D points residing on a single plane (specifically for 3D-2D alignment).

To achieve this, we leverage the progressive Poisson Sampling technique [7]. Commencing with an initially large disk radius R, we search for four samples that maintain a minimum separation of R units. If these samples satisfy the non-degeneracy criteria, the process is successful. If not, we reduce R by one-tenth and repeat the procedure until a suitable set of samples is obtained or until the process fails. Figure 6 visually depicts an instance of point selection within one of the RGB frames using this method.

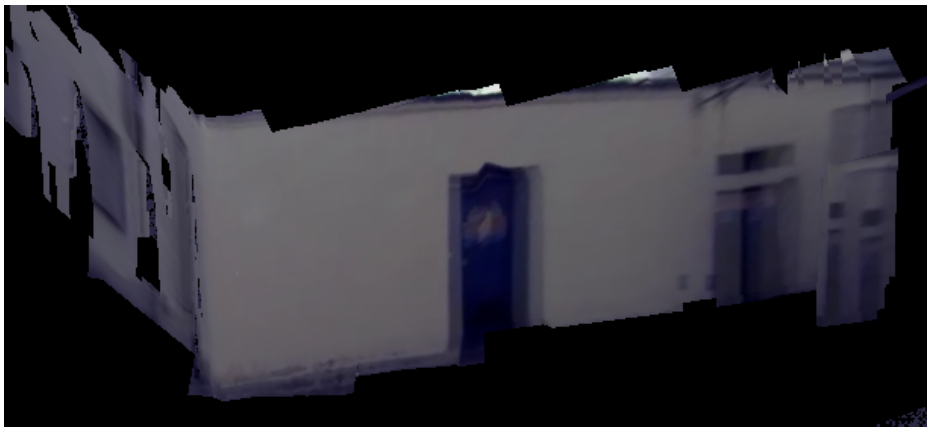

Fig. 7: Two images that overlap imprecisely create the illusion of fuzzy discontinuities, like at the frame of the blue doors in the image.

4.3 Calibration refinement

As previously mentioned, the asynchronous collection of point clouds and images can adversely affect the quality of the calibration procedure by providing imprecise 3D-2D correspondences. This, in turn, leads to the occurrence of ghosting artifacts, which become apparent when multiple images of the same geometry are not precisely aligned (Figure 7 illustrates an example). To enhance the accuracy of the registration process and minimize these ghosting artifacts, we propose an output-sensitive procedure.

Our algorithm operates in pairs. It designates one camera as the reference and adjusts the extrinsic parameters of the other camera (i.e., position and orientation) to minimize ghosting. We assess the extent of ghosting using a measure of photo-consistency, which evaluates the similarity in color assigned by the two cameras to the same 3D surface points.

Fig. 8: The scheme shows the reference camera F and the camera to be fine-aligned (M). The images are the actual feed from the physical camera and the rendering with the current extrinsic parameters.

Figure 8 illustrates a common scenario in which two camera frusta overlap. In this context, we will refer to one camera as the 'Fixed' camera (denoted as F), which serves as the reference camera, and the other as the 'Moving' camera (denoted as M), which we aim to align with the fixed camera. We use the notation $I(.)$ to represent the images captured by each respective camera. Both cameras have a partial view of the environment, and their views intersect within a region denoted as R (highlighted in blue in the figure).

Our approach involves texturing the region R by projecting the image $I(F)$ onto it, and subsequently rendering this textured region R from the perspective of camera M. We will denote the set of pixels covered by the projection of R as R_p. In an ideal scenario where the geometry is accurate, and the cameras are perfectly aligned, the projection of the textured scene onto camera M should perfectly match the actual image $I(M)$ within the same portion of the image or screen (assuming also perfectly Lambertian material).

To quantify the disparity between the rendered image and the actual image, we define an error function. This error function serves as a measure of how different the rendering is from the actual image, allowing us to initiate a minimization algorithm. This minimization algorithm adjusts the extrinsic parameters of camera M as variables in order to improve the alignment between the two cameras.

We define our error function using a pixel-to-pixel color difference restricted to R_p, that is, without considering the pixels not included in the rendering. Naturally, such projection changes every time the extrinsic parameters are updated but we are working on the assumption that this is a refinement of an existing camera registration and hence that said change of R_p will be small. Therefore we compute the initial projection and create a mask by dilating it by a certain number of pixels (50 in our experiments). This is a way to restrict the amount of camera movement in a neighborhood of the initial solution. Doing this will tend to avoid local minima of the error function that would be found if R_p could be in every portion of the screen. Please note that bounding the projection can be seen as the output-sensitive alternative to directly apply upper and lower bounds to the parameters (i.e. camera position and rotation).

The exact definition of our error function is given in Equation 1. N is the total number of pixels, i and j identify the row and column of a pixel position, $Mask(i, j)$ is 1 if the pixel i, j is in the dilated region aforementioned, $colorDist(i, j)$ is the distance between corresponding pixels in the two images and th is a threshold value. Note that the sum is counting how many corresponding pixels in the masked region are closer than a threshold in color space. Hence, the error function (Err) spans the range $[N - N_m, N]$, with N_m representing the count of masked pixels, within a total of N pixels.

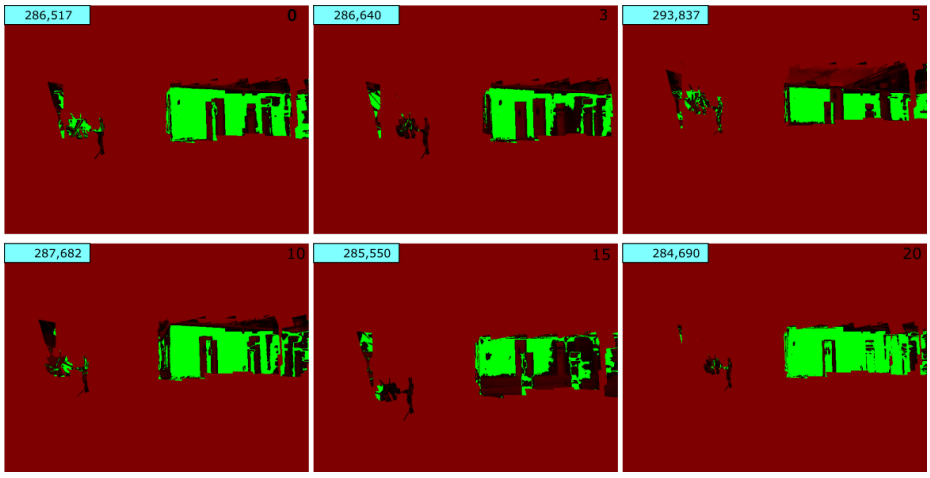

Fig. 9: A sampling of the iterations of the minimization procedure. The function $colorDist$ is mapped to the red channel, if the value is below a given threshold (0.2 in this experiment) the pixel is rendered green.

We tested several derivative-free minimization algorithms for Non-Linear problems and found the most consistent results with the NEWUOA [32] implementation provided by the C++ library NLOpt [15] The evaluation of $Err(M)$ is carried out by harnessing the graphics hardware. We use a fullscreen quad and provide the image $I(M)$, the result of the rendering from M, and the mask. The fragment shader computes colorDist and *discards* a fragment if the value is below the threshold and the mask value is set 1. On the client side, we use the OpenGL *occlusion query* mechanism to count how many fragments passed the depth test, which in our case means how many where *not* discarded, which turns out to be our Err function. Figure 10 show the same geometry as Figure 7 after the fine registration step.

$$\begin{aligned} \mathrm{Err(M)} &= N - \sum_{\forall i,j} \mathrm{dist}(i,j) \times \mathrm{Mask}(i,j) \\ \mathrm{dist}(i,j) &= \begin{cases} 1 & \text{if colorDist}(i,j) \leq \mathrm{th} \\ 0 & \text{otherwise} \end{cases} \end{aligned} \tag{1}$$

Figure 9 shows some steps of the minimization process. The color of pixel (i, j) encodes the value colorDist(i, j) on the red channel (remapped between 0 and 0.5) if it is greater than the threshold, and as green if it is lower than the threshold and if Mask$(i, j) == 1$. Therefore, the amount of green pixels is the result of the summation in Equation 1.

Fig. 10: A snapshot of the application with the same geometry as in Figure 7 after the fine registration step.

This procedure should be regarded as a supplementary tool aimed at enhancing registration through point-to-point correspondences, as described in Section 4.2. Various scene-dependent parameters influence the final outcome.

One critical factor is the degree of dilation applied to the projection, denoted as R_p. If this value is too large, the minimization process might become trapped

in a local minimum. Conversely, if it is too small, there may be insufficient improvement over the initial values.

Another crucial consideration is the precise definition of $ColorDist$. We currently employ the difference in the CIELab color space. However, it's important to note that this implicitly assumes that materials exhibit mostly Lambertian properties, which may not always hold true.

Additionally, the threshold value Th wields significant influence. Setting it too high may result in plateaus in the error function, effectively treating every pixel as a good matching one. Conversely, setting it too low can lead to a discontinuous error function, potentially causing local minima.

5 REAL-TIME ACQUISITION AND RENDERING FROM LIDAR AND RGB CAMERAS.

In our experiments, we tailored the comprehensive system outlined in Section 3 to enable real-time acquisition and rendering of multi-sensor data. LIDARs and camera feeds are combined at run-time to offer a free point-of-view rendering. This real-time rendering is achieved through immediate point cloud tessellation and projective mapping, as elaborated in the subsequent explanation.

Tessellation of the point clouds The challenge of defining a 2D surface from a point cloud has a long history with various proposed solutions, as outlined in a comprehensive survey by [2]. The inherent complexities of this problem, such as sparsity and irregular sampling, often require the solver to make assumptions about the nature of the sampled surface. However, in the case of LIDARs, where sampling is partially dense and structured on a grid, it becomes both feasible and reasonable to employ a predefined regular tessellation of the points, simplifying the surface reconstruction process. Note that all of the above does not require any processing on the CPU side, the data arriving from the LIDARs are sent to the GPU as vertices, the tessellation pattern is static and for filtering the triangles we use a geometry shader that discards the unwanted triangles.

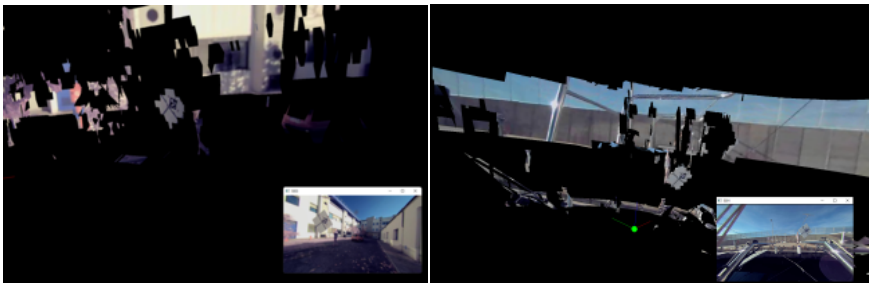

Fig. 11: Snapshots of the textured geometry with the camera feeds shown at their respective bottom right (from [9]).

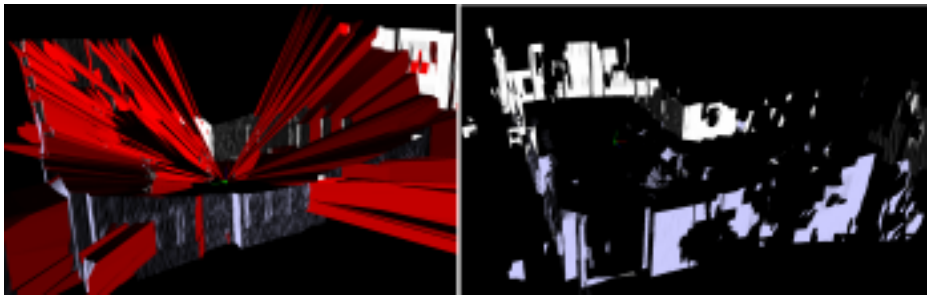

Fig. 12: Elongated triangles (in red) usually connect portions of surfaces that are disconnected/different and hence automatically removed. The Left and right images show the same data before and after the removal of triangles (from [9]).

Projecting RGB images The final geometry's color is determined through projective texturing [24]. This involves an initial rendering pass in which shadow maps are created for each camera and then utilized in the subsequent rendering pass to ascertain the visibility of each point from different camera perspectives. It is important to note that certain regions in space may be covered by multiple cameras. Consequently, an efficient method is required to merge their often distinct contributions effectively. In our approach, inspired by the work in [5], we blend the contributions of each camera based on the cosine of the angle between the projection direction and the surface normal. Additionally, we enhance image blending in overlapping regions through an image equalization process utilizing histogram matching [31].

Remote Rendering A realistic rendering of the environment is achieved through the tessellated point cloud, followed by projective texturing based on the camera's vantage point. To maintain the quality of the rendered output, rendering data is directly retrieved from the GPU's framebuffer and processed as a stream of JPEG images. The remote rendering process is enabled through a dedicated MJPEG streaming network socket connection, separate from our standard client-server communication. These MJPEG-encoded videos are then transmitted over HTTP protocols, using a highly adaptable, multi-threaded, and computationally efficient streaming framework. This setup allows clients to remotely view real-time rendered content directly in a web browser. In addition to MJPEG streaming, our application provides support for visualizing the rendered content as H.264 streams over the Real-Time Streaming Protocol (RTSP) using the FFMPEG library [40].

6 CONCLUSION

We introduced an improved version of the system presented in [9] for real-time acquisition and rendering of 3D scenes using low-cost LIDARs and RGB cameras, dubbed NausicaaVR. NausicaaVR supports semi-automatic calibration,

on-the-fly tessellation, and remote rendering, and it is built with a client-server architecture.

While our system's results were satisfactory for environmental awareness, challenges remain to achieve immersive real-time rendering. One concern is the precise alignment of data. Even with the improvement proposed in Section 4.3 of this paper, spatiotemporal calibration of image and geometry remains sensitive to the specific scene and sensor placements.

One approach worth considering is to project images onto the geometry, even if there's a slight misalignment, and then utilize learning algorithms to minimize or eliminate the ghosting effect. This technique has been employed successfully in related domains like denoising [39] and deblurring [44]. Another obstacle to address is the absence of geometry in areas not captured by LIDAR. These unscanned regions of the scene result in certain parts of the images being projected onto the background as if they were located on the horizon.

One option is to use image segmentation and classification to find parts that are not at the horizon and avoid projecting those pixels if there is no corresponding geometry. This avoids the unpleasant anamorphic effect of misprojection but leaves holes in the rendering. A more intriguing approach is to use image-based techniques to complete the geometry from the existing sampling. In other words, using simple proxy 3D elements for projecting the images. This method would also require precise image segmentation of the images to avoid pixels belonging to the background (e.g., the sky) being projected into such impostors.

References

1. An, P., Ma, T., Yu, K., Fang, B., Zhang, J., Fu, W., Ma, J.: Geometric calibration for lidar-camera system fusing 3d-2d and 3d-3d point correspondences. Opt. Express **28**(2), 2122–2141 (Jan 2020). https://doi.org/10.1364/OE.381176, https://opg.optica.org/oe/abstract.cfm?URI=oe-28-2-2122
2. Berger, M., Tagliasacchi, A., Seversky, L.M., Alliez, P., Guennebaud, G., Levine, J.A., Sharf, A., Silva, C.T.: A survey of surface reconstruction from point clouds. In: Computer Graphics Forum. vol. 36, pp. 301–329. Wiley Online Library (2017)
3. Brinkmann, M., Hahn, A.: Testbed architecture for maritime cyber physical systems. In: 2017 IEEE 15th International Conference on Industrial Informatics (INDIN). pp. 923–928 (2017). https://doi.org/10.1109/INDIN.2017.8104895
4. Bu, Z., Sun, C., Wang, P., Dong, H.: Calibration of camera and flash lidar system with a triangular pyramid target. Applied Sciences **11**(2) (2021). https://doi.org/10.3390/app11020582, https://www.mdpi.com/2076-3417/11/2/582
5. Callieri, M., Cignoni, P., Corsini, M., Scopigno, R.: Masked photo blending: mapping dense photographic dataset on high-resolution 3d models. Computer & Graphics **32**(4), 464–473 (Aug 2008), http://vcg.isti.cnr.it/Publications/2008/CCCS08, for the online version: http://dx.doi.org/10.1016/j.cag.2008.05.004
6. Chen, Q., Xie, Y., Guo, S., Bai, J., Shu, Q.: Sensing system of environmental perception technologies for driverless vehicle: A review of state of the art and challenges. Sensors and Actuators A: Physical **319**, 112566 (2021). https://doi.org/https://doi.org/10.1016/j.sna.2021.112566, https://www.sciencedirect.com/science/article/pii/S0924424721000273

7. Corsini, M., Cignoni, P., Scopigno, R.: Efficient and flexible sampling with blue noise properties of triangular meshes. IEEE Transactions on Visualization and Computer Graphics **18**(6), 914–924 (2012). https://doi.org/10.1109/TVCG.2012.34
8. dspace: https://www.dspace.com/en/pub/home/applicationfields/stories/smartkai-parking-assistance-f.cfm (2021)
9. Dutta, S., Ganovelli, F., Cignoni, P.: On-the-fly acquisition and rendering with low cost lidar and RGB cameras for marine navigation. In: Grueau, C., Rodrigues, A., Ragia, L. (eds.) Proceedings of the 9th International Conference on Geographical Information Systems Theory, Applications and Management, GISTAM 2023, Prague, Czech Republic, April 25-27, 2023. pp. 176–183. SCITEPRESS (2023). https://doi.org/10.5220/0011855000003473, https://doi.org/10.5220/0011855000003473
10. Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F., Marín-Jiménez, M.: Automatic generation and detection of highly reliable fiducial markers under occlusion. Pattern Recognition **47**(6), 2280–2292 (2014). https://doi.org/https://doi.org/10.1016/j.patcog.2014.01.005, https://www.sciencedirect.com/science/article/pii/S0031320314000235
11. Gong, X., Lin, Y., Liu, J.: 3d lidar-camera extrinsic calibration using an arbitrary trihedron. Sensors **13**(2), 1902–1918 (2013), https://www.mdpi.com/1424-8220/13/2/1902
12. Grammatikopoulos, L., Papanagnou, A., Venianakis, A., Kalisperakis, I., Stentoumis, C.: An effective camera-to-lidar spatiotemporal calibration based on a simple calibration target. Sensors **22**(15) (2022), https://www.mdpi.com/1424-8220/22/15/5576
13. Hahn, T., Damerius, R., Rethfeldt, C., Schubert, A.U., Kurowski, M., Jeinsch, T.: Automated maneuvering using model-based control as key to autonomous shipping. at - Automatisierungstechnik **70**(5), 456–468 (2022). https://doi.org/doi:10.1515/auto-2021-0146, https://doi.org/10.1515/auto-2021-0146
14. ImagingSource: https://www.theimagingsource.com (2017)
15. Johnson, S.G.: The NLopt nonlinear-optimization package. https://github.com/stevengj/nlopt (2007)
16. Kang, J., Doh, N.L.: Automatic targetless camera–lidar calibration by aligning edge with gaussian mixture model. Journal of Field Robotics **37**(1), 158–179 (2020). https://doi.org/https://doi.org/10.1002/rob.21893, https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21893
17. Kim, H., Kim, D., Park, B., Lee, S.M.: Artificial intelligence vision-based monitoring system for ship berthing. IEEE Access **8**, 227014–227023 (2020). https://doi.org/10.1109/ACCESS.2020.3045487
18. Kodaira, A., Zhou, Y., Zang, P., Zhan, W., Tomizuka, M.: Sst-calib: Simultaneous spatial-temporal parameter calibration between lidar and camera. p. 2896–2902. IEEE Press (2022). https://doi.org/10.1109/ITSC55140.2022.9922085, https://doi.org/10.1109/ITSC55140.2022.9922085
19. Li, Q., Queralta, J.P.n., Gia, T.N., Zou, Z., Westerlund, T.: Multi-sensor fusion for navigation and mapping in autonomous vehicles: Accurate localization in urban environments. Unmanned Systems **08**(03), 229–237 (2020). https://doi.org/10.1142/S2301385020500168, https://doi.org/10.1142/S2301385020500168

20. Li, X., He, F., Li, S., Zhou, Y., Xia, C., Wang, X.: Accurate and automatic extrinsic calibration for a monocular camera and heterogenous 3d lidars. IEEE Sensors Journal **22**(16), 16472–16480 (2022). https://doi.org/10.1109/JSEN.2022.3189041
21. Martelli, M., Virdis, A., Gotta, A., Cassarà, P., Di Summa, M.: An outlook on the future marine traffic management system for autonomous ships. IEEE Access **9**, 157316–157328 (2021). https://doi.org/10.1109/ACCESS.2021.3130741
22. Moghadam, P., Bosse, M., Zlot, R.: Line-based extrinsic calibration of range and image sensors. In: 2013 IEEE International Conference on Robotics and Automation. pp. 3685–3691 (2013). https://doi.org/10.1109/ICRA.2013.6631095
23. Nowicki, M.R.: Spatiotemporal calibration of camera and 3d laser scanner. IEEE Robotics and Automation Letters **5**, 6451–6458 (2020)
24. NVidia: https://www.nvidia.com/en-us/drivers/Projective-Texture-Mapping/ (2001)
25. Nvidia: Nvidia announces jetson tx2: Parker comes to nvidia's embedded system kit (2017)
26. Paneque, J., Valseca, V., Martínez-de Dios, J.R., Ollero, A.: Autonomous reactive lidar-based mapping for powerline inspection. In: 2022 International Conference on Unmanned Aircraft Systems (ICUAS). pp. 962–971 (2022). https://doi.org/10.1109/ICUAS54217.2022.9836213
27. Park, C., Moghadam, P., Kim, S., Sridharan, S., Fookes, C.: Spatiotemporal camera-lidar calibration: A targetless and structureless approach. IEEE Robotics and Automation Letters **5**(2), 1556–1563 (2020). https://doi.org/10.1109/LRA.2020.2969164
28. Park, Y., Yun, S., Won, C.S., Cho, K., Um, K., Sim, S.: Calibration between color camera and 3d lidar instruments with a polygonal planar board. Sensors **14**(3), 5333–5353 (2014). https://doi.org/10.3390/s140305333, https://www.mdpi.com/1424-8220/14/3/5333
29. Perera, L., Moreira, L., Santos, F., Ferrari, V., Sutulo, S., Soares, C.G.: A navigation and control platform for real-time manoeuvring of autonomous ship models. IFAC Proceedings Volumes **45**(27), 465–470 (2012). https://doi.org/https://doi.org/10.3182/20120919-3-IT-2046.00079, https://www.sciencedirect.com/science/article/pii/S1474667016312733, 9th IFAC Conference on Manoeuvring and Control of Marine Craft
30. Peršić, J., Petrović, L., Marković, I., Petrović, I.: Spatiotemporal multisensor calibration via gaussian processes moving target tracking. IEEE Transactions on Robotics **37**(5), 1401–1415 (2021). https://doi.org/10.1109/TRO.2021.3061364
31. Pizer, S.M., Amburn, E.P., Austin, J.D., Cromartie, R., Geselowitz, A., Greer, T., ter Haar Romeny, B., Zimmerman, J.B., Zuiderveld, K.: Adaptive histogram equalization and its variations. Computer vision, graphics, and image processing **39**(3), 355–368 (1987)
32. Powell, M.J.D.: The NEWUOA software for unconstrained optimization without derivatives. In: Pillo, G.D., Roma, M. (eds.) Large-Scale Nonlinear Optimization, Nonconvex Optimization and Its Applications, vol. 83, pp. 255–297. Springer (2006)
33. Rehder, J., Beardsley, P., Siegwart, R., Furgale, P.: Spatio-temporal laser to visual/inertial calibration with applications to hand-held, large scale scanning. In: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 459–465 (2014). https://doi.org/10.1109/IROS.2014.6942599
34. Rüssmeier, N., Hahn, A., Nicklas, D., Zielinski, O.: Ad-hoc situational awareness by optical sensors in a research port maritime environment , approved networking and sensor fusion technologies (2016)

35. Schiaretti, M., Chen, L., Negenborn, R.R.: Survey on autonomous surface vessels: Part i - a new detailed definition of autonomy levels. In: Bektaş, T., Coniglio, S., Martinez-Sykora, A., Voß, S. (eds.) Computational Logistics. pp. 219–233. Springer International Publishing, Cham (2017)
36. Schubert, A.U., Kurowski, M., Gluch, M., Simanski, O Jeinsch, T.: Manoeuvring automation towards autonomous shipping. Zenodo (Oct 2018). https://doi.org/10.24868/issn.2631-8741.2018.020, https://doi.org/10.24868/issn.2631-8741.2018.020
37. Snyder, F.D., Morris, D.D., Haley, P.H., Collins, R.T., Okerholm, A.M.: Autonomous river navigation. In: SPIE Optics East (2004)
38. Thombre, S., Zhao, Z., Ramm-Schmidt, H., Vallet García, J.M., Malkamäki, T., Nikolskiy, S., Hammarberg, T., Nuortie, H., H. Bhuiyan, M.Z., Särkkä, S., Lehtola, V.V.: Sensors and ai techniques for situational awareness in autonomous ships: A review. IEEE Transactions on Intelligent Transportation Systems **23**(1), 64–83 (2022). https://doi.org/10.1109/TITS.2020.3023957
39. Tian, C., Fei, L., Zheng, W., Xu, Y., Zuo, W., Lin, C.W.: Deep learning on image denoising: An overview. Neural Networks **131**, 251–275 (Nov 2020). https://doi.org/10.1016/j.neunet.2020.07.025, https://linkinghub.elsevier.com/retrieve/pii/S0893608020302665
40. Tomar, S.: Converting video formats with ffmpeg. Linux Journal **2006**(146), 10 (2006)
41. Tonnis, M., Lindl, R., Walchshausl, L., Klinker, G.: Visualization of spatial sensor data in the context of automotive environment perception systems. In: 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality. pp. 115–124 (2007). https://doi.org/10.1109/ISMAR.2007.4538835
42. Vu, T.D., Aycard, O., Tango, F.: Object perception for intelligent vehicle applications: A multi-sensor fusion approach. In: 2014 IEEE Intelligent Vehicles Symposium Proceedings. pp. 774–780 (2014). https://doi.org/10.1109/IVS.2014.6856588
43. Wright, R.G.: Intelligent autonomous ship navigation using multi-sensor modalities. TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation **13**(3), 503–510 (2019). https://doi.org/10.12716/1001.13.03.03
44. Zhang, K., Ren, W., Luo, W., Lai, W.S., Stenger, B., Yang, M.H., Li, H.: Deep Image Deblurring: A Survey. International Journal of Computer Vision **130**(9), 2103–2130 (Sep 2022). https://doi.org/10.1007/s11263-022-01633-5, https://link.springer.com/10.1007/s11263-022-01633-5