The Visual Computer (2024) 40:571–584
https://doi.org/10.1007/s00371-023-02802-4

ORIGINAL ARTICLE

Embedded 3D reconstruction of dynamic objects in real time for maritime situational awareness pictures

Felix Sattler[1] · Borja Carrillo-Perez[1] · Sarah Barnes[1] · Karsten Stebner[2] · Maurice Stephan[1] · Gregor Lux[3]

Accepted: 8 February 2023 / Published online: 10 March 2023

Abstract
Assessing the security status of maritime infrastructures is a key factor for maritime safety and security. Facilities such as ports and harbors are highly active traffic zones with many different agents and infrastructures present, like containers, trucks or vessels. Conveying security-related information in a concise and easily understandable format can support the decision-making process of stakeholders, such as port authorities, law enforcement agencies and emergency services. In this work, we propose a novel real-time 3D reconstruction framework for enhancing maritime situational awareness pictures by joining temporal 2D video data into a single consistent display. We introduce and verify a pipeline prototype for dynamic 3D reconstruction of maritime objects using a static observer and stereoscopic cameras on an GPU-accelerated embedded device. A simulated dataset of a harbor basin was created and used for real-time processing. Usage of a simulated setup allowed verification against synthetic ground-truth data. The presented pipeline runs entirely on a remote, low-power embedded system with ~6 Hz. A Nvidia Jetson Xavier AGX module was used, featuring 512 CUDA-cores, 16 GB memory and an ARMv8 64-bit octa-core CPU.

Keywords Situtational awareness · Dynamic 3D reconstruction · Real-time monitoring · Maritime safety and security

✉ Felix Sattler
felix.sattler@dlr.de

Borja Carrillo-Perez
borja.carrilloperez@dlr.de

Sarah Barnes
sarah.barnes@dlr.de

Karsten Stebner
karsten.stebner@dlr.de

Maurice Stephan
maurice.stephan@dlr.de

Gregor Lux
gregor.lux@w-hs.de

1 Maritime Security Technologies, Institute for the Protection of Maritime Infrastructures, German Aerospace Center (DLR), Fischkai 1, 27572 Bremerhaven, Bremen, Germany

2 Security Research and Applications, Institute of Optical Sensor Systems, German Aerospace Center (DLR), Rutherfordstr. 2, 12489 Berlin, Berlin, Germany

3 Department of Computer Science, Westphalian University of Applied Sciences, Neidenburger Str. 43, 45897 Gelsenkirchen, NRW, Germany

1 Introduction

Maritime transport is a crucial element of global trading, helping to facilitate a close interdependency between countries, manufacturers and markets [1]. Maritime infrastructures such as harbor areas and cargo terminals, are critical for the successful functioning of the maritime transport chains. Therefore, monitoring their security, integrity and operational safety is of key importance. This work focuses on the improvement of optical maritime infrastructure monitoring by detecting and reconstructing dynamic maritime objects for a consistent 3D display.
We present a novel system prototype that performs real-time 3D reconstruction of dynamic maritime objects in static scenes using stereoscopic cameras on an embedded system. The use of a GPU-accelerated embedded system is a cost-effective solution that allows our pipeline to run in situ at remote locations. Our pipeline is able to generate consistent 3D point clouds from video data in real time, reducing the required bandwidth for transmission and improving spatial information. Real time in this context is defined with respect to speed limits in ports in Germany as introduced by the Fed-

eral Waterways and Shipping Agency (GDWS).[1] A ship is allowed to have a maximum speed of 10–15 kt which is a displacement of ~5 to 8 m/s. To compensate for large shifts in position and transformation the pipeline must therefore run as fast as possible. With the current setup, we compensate displacements of ~0.85 m to ~1.28 m at ~6 Hz.

Maritime dynamic objects are targets of observation that navigate through and interact with the maritime infrastructure. In our work, we focus on ships though our system prototype can be readily expanded to include more categories like cars, trucks or cargo. The term *static scene* describes the non-moving, rigid-body structures and refers to the maritime infrastructure itself.

A vast body of research exists for indoor, small-scale RGB-D[2] systems in the context of autonomous robotics [2] as well as on LiDAR[3] and stereoscopic outdoor 3D reconstruction for autonomous driving [3,4]. Due to the sensor setups for moving observers as shown in [5], a specific motion model is implicitly assumed. In our use case, which deals with static camera views, no moving observer is present and large scenes with moving water bodies are considered which limit the use of these existing works.

The presented system is a modular pipeline for GPU-accelerated embedded devices and stereo cameras. Using a multistage approach, we first:

1. Detect ships in input images using AI methods
2. Segment them using the objects motion information
3. Perform depth matching using the stereo images
4. Register current depth information with previous information to obtain a transformation matrix
5. Integrate the current depth data using the calculated transformation into a dense volumetric representation

As a proof of concept, the pipeline presented in this work was tested and verified using a 3D simulation in a virtual, georeferenced environment for a single dynamic object only. We achieve a runtime of ~6 Hz while also producing a dense point cloud that resembles the shape of the target ship and is robust against outliers. Even with a significant amount of drift in translation compared to the ground-truth, the system still manages to obtain a complete and readable reconstruction of the ship.

[1] https://www.gdws.wsv.bund.de/SharedDocs/Pressemitteilungen/DE/20190116_Geschwindigkeitsbegrenzung_Elbe_PM.html, Accessed 18 January 2023.

[2] RGB-D: RGB camera with short-range, active depth sensors.

[3] LiDAR: Light detection and ranging.

2 Related work

Dynamic 3D reconstruction can be split into several subproblems with their respective research bodies associated with them. Most of the related work stems from the domains of autonomous driving and robotics where simultaneous localization and mapping (SLAM) systems that employ 3D reconstruction are the main focus of research. However, it is outlined how these approaches are fundamentally different from our proposed system. Therefore, this section will focus on complete, state-of-the-art 3D reconstruction systems only.

Newcombe et al. [6] proposed a complete system that can be considered a basis of real-time 3D reconstruction. Their KinectFusion system can perform stereo-based 3D reconstruction in static environments using RGB-D cameras in real time (~2 Hz). Grinvald et al. [7] proposed improvements to this work through the incorporation of auxiliary data like multiple objects and color data. A core aspect of [6] is the use of truncated signed distance fields (TSDF) and a volumetric grid to fuse depth information into a consistent, implicit representation. TSDFs can be generated through projective data association [8]. Using a projection to associate depth data makes the samples linearly independent and allows parallel computing on a GPU device. Since the technique by [8] does not create true Euclidean distance fields, the methods requires different viewing angles for robust convergence. This makes the method suitable for applications which rely on motion for 3D reconstruction like the framework presented here.

Whelan et al. [2] adopted the framework by Newcombe et al. [6] and proposed a large-scale SLAM system in the context of robotics. They improved on the computational performance of object tracking, allowing for faster motions and shorter update increments. Moreover, their system added a streaming approach to support larger environments and higher spatial resolution. A major contribution is the development of a highly efficient variant of the iterative-closest point algorithm (ICP) for GPU devices that allows real-time tracking of point clouds. This work, however, performs static 3D reconstruction using handheld RGB-D cameras as a means to generate 3D static maps of indoor areas. Since RGB-D cameras employ active illuminators in the near infrared spectrum for increased depth resolution, they cannot be used in outdoor environments with strong infrared lighting.

Therefore, a different system is required to infer good depth estimates. This was addressed by [4] who proposed a complete stereo-based hybrid real-time 3D reconstruction system. Developed in the context of autonomous driving, their system achieves ~5 Hz on desktop hardware. The system can perform 3D reconstruction of static environments and dynamic objects. Still, the system does not use embedded systems and aims to generate a 3D static map. Also, it assumes a moving observer (camera) that is positioned at a fixed height in the direction of travel. This assumption simplifies the motion model because the ego-motion of the camera is smooth and always relative to the heading of the car (limiting the degrees of freedom). Also, the system only works in close range where the depth error is limited, approaching and passing objects of interest throughout the test sequences to smooth out noise. The work also addresses the concept of noise filtering (voxel garbage collection) in the output of the voxel volume based on [9]. This is an important factor to consider when using reconstructed data for display.

Regarding the segmentation of dynamic agents, [10] evaluated several techniques for tracking and masking. While not being a direct 3D reconstruction system, their work outlines instance-aware object tracking by using 3D reconstructed data as input. While the system performs online, it is non-real time, again with a focus on desktop hardware. The proposed system can identify several classes of objects but is also tested and verified in the context of autonomous driving. Thus, the same restrictions as in [4] apply. Nevertheless, results have low error showing how 3D reconstruction can help leveraging object tracking. While the technique for depth estimation is based on SfM (structure-from-motion), the basic principle can be readily transferred to other algorithms.

A full dynamic 3D reconstruction system using instance-aware tracking was presented by [11]. They propose a hybrid 3D reconstruction system that, similarly to Barsan et al. [4], can work with static environments and dynamic agents. Their main contribution is the inclusion of several TSDF volumes for different dynamics agents and the static environment. Instance segmentation to isolate dynamic objects is performed using the well-known Mask R-CNN [12]. Their probabilistic framework allows for detailed integration results for small-scale indoor environments. However, performance on a desktop GPU and a high-end CPU is limited to ∼4 to 9 Hz showing the high-performance demand. The largest performance drop stems from the use of Mask R-CNN@.

LiDAR technology is also researched in great detail. The spatial sparsity of stationary LiDAR together with noisy water surface interaction as addressed by [13] leads to distortions and the necessity of extensive filtering. Therefore, techniques that rely on LiDAR data for 3D reconstruction are not in the scope of this work.

Our pipeline was developed as a modular multistage system like [4], supporting multi-object reconstruction similar to [7,11] with a unique volume per object. Instead of using deep learning for instance segmentation as [10,11], we use a state-of-the-art object detector and segment its estimated bounding box with an approximation based on motion to allow our pipeline to run at a higher framerate on the embedded systems. Since a clean display is important for our use case, the noise reduction techniques presented by [4,9] are extended, implemented and verified in our presented pipeline.

3 Dataset creation

To validate our pipeline, we created a virtual dataset in the open-source 3D software Blender3D [14]. The software was chosen because it offers a complete set of tools from 3D modeling, virtual scene layout, animation and simulation as well as photorealistic image synthesis (rendering). It has been successfully used in other works to create simulation frameworks for machine learning and computer vision tasks [15–17]. Besides Blender 3D other work has shown that a similar procedure can be realized in a game engine like Unreal Engine [18].

Since we wanted to have ground-truth references for all pipeline stages, we digitally re-created a physical harbor basin to scale including physically based materials and rendering. Figure 1 shows a comparison between the real harbor basin with different boats in frame and the digital copy with the tugboat. Both images use the same calibrated camera and an identical perspective. Note that due to the removal of reference points the perspective looks slightly different. However, the scene was created using a geo-referenced digital 3D map of the harbor area. Depth is between ∼25 and ∼60 m, motivated by typical observation scenarios at port basins and locks. To verify our dynamic 3D reconstruction system, we used a virtual re-creation of a real tugboat to scale (∼35 m length). A tugboat is a common boat found at port infrastructures. Also, its size allowed for a got fit into the 3D scene we virtually re-created.

It is important to outline that the simulated sequence does not take the physical interaction between vessel and water into account. Since we are dealing with large-scale water bodies a fluid simulation that allows for dynamic wake and whitewater in an image sequence is complex and was outside the scope of this work. The static images that we used for training our detection dataset (as outlined in Sect. 4.1) feature wake and whitewater. We focused on producing highly realistic results to minimize the bias when compared to real-world images. In return, we found that the object detectors still detect ships without wake and whitewater. Still, the technique used here was not stable in image sequences and lacked temporal resolution.

The sequence is 25 s long (624 frames) and synthesizing (rendering) the scene took several minutes per frame on an Intel Xeon E5-2683 v4 CPU with 64 cores running at 2.1 GHz, 64 GB memory and a Nvidia Quadro GV100 with 32 GB and 5120 compute units (CUDA-cores). It was created as a stereoscopic dataset with a distance of 4 m between cameras. While this is a very large baseline with respect to most stereo systems, Yang and Lu [19] propose in their work a system for tsunami detection that uses a 27 m baseline. Besides stereoscopic frames, the dataset contains ground-truth values for every pipeline stage including 3D shape, object transformation, camera geo-reference (UTM projec-

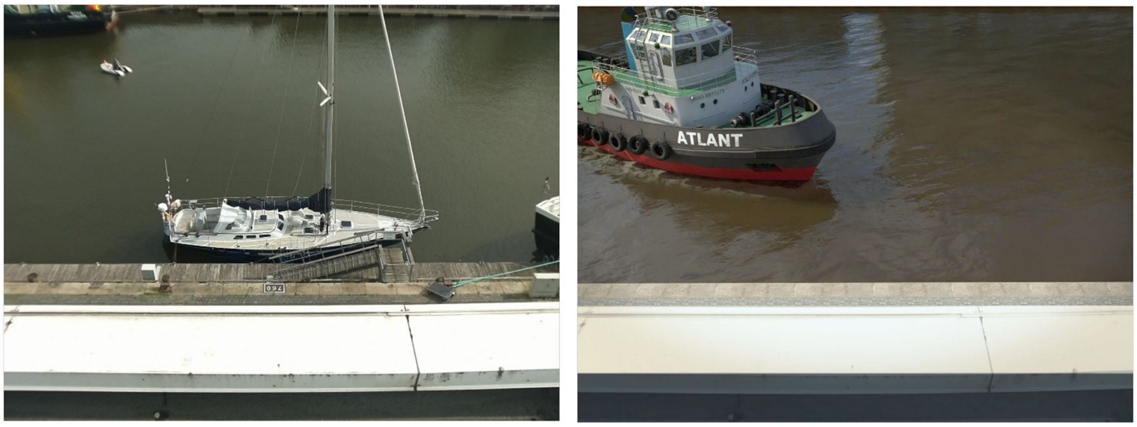

Fig. 1 A comparison between the real harbor basin (left) and the digital re-creation (right). Notable are the accuracy of reflections on the water surface and materials. Please note that both images have the same camera parameters and perspective

tion), depth information (1cm resolution), mask information and a bounding box.

4 Dynamic embedded 3D reconstruction

Real-time dynamic 3D reconstruction is performed on a GPU-accelerated embedded device, the Nvidia Jetson Xavier AGX, including 512 compute units (CUDA-cores), 16 GB memory and an ARMv8 64-bit octa-core CPU. It also has a dedicated coprocessor for low-level vision tasks (refer to Sect. 4.2).[4] Processing is done by a pipeline comprised of several successive stages. All computations are done on a per frame basis except the last stage, which is point cloud extraction. This is only done on demand when a new representation is requested. All computations including preprocessing are done on the embedded device. Figure 2 shows an overview of the processing pipeline, from left to right:

1. Stereoscopic camera frames are preprocessed to prepare a pipeline run. Resampling, color channel conversion and copying are performed using GPU acceleration.
2. Object detection is performed using a CNN on the left stereo frame to generate a 2D bounding box.
3. Dense optical flow vectors are computed for the left frame, converted to a binary mask and multiplied by the 2D bounding box to segment the left frame.
4. A dense depth map is created from stereo image pairs and multiplied with the previously calculated binary mask to filter out unwanted background information.
5. Using the masked depth map, object tracking is performed to estimate the visual odometry (i.e., the motion estimation) of the dynamic object.
6. Tracking information, depth map and left color frame are fused into a consistent volumetric representation.
7. Output of the pipeline is a filtered 3D point cloud with adaptive voxel garbage collection.

Each stage will be briefly described in the following sections to provide an overview of the algorithms and techniques involved. The preprocessing stage uses common computer vision techniques and is therefore not discussed.

4.1 Object detection

In order to extract the position of a maritime object, a robust and near real-time object detector is needed. The object detector used in this work must provide the bounding box of a ship with minimum possible inference time. Object detection is always performed on the left stereo image. For this work, the YOLOv5 method is selected [20]. It is a suitable algorithm for real-time object detection trained on the MS COCO dataset [21] that can perform efficiently on an embedded system. The algorithm was deployed on embedded device using the Pytorch framework [22]. For a robust ship detection model, we rendered an additional training set of 114 images of a tugboat in different sizes and perspectives that are used for further training of YOLOv5. In Fig. 3, some samples of the additional training set and an example of the tugboat detection on our virtual stereo dataset (see Sect. 3) are shown. If no object was detected during this stage, the pipeline starts over again.

[4] https://siliconhighway.com/wp-content/gallery/jetson-agx-xavier-developer-kit-datasheet-us-811514-r5-web.pdf, Accessed 18 January 2023.

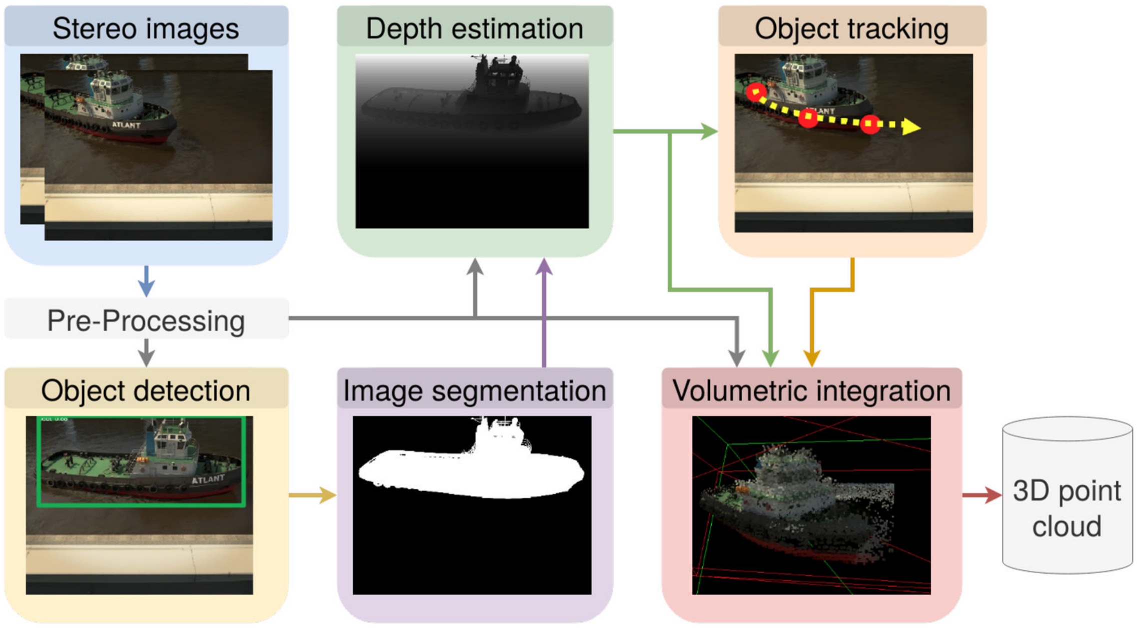

Fig. 2 A schematic view of the reconstruction pipeline with ground-truth examples for each stage. Note that for the object tracking stage the dots are positions in time and the yellow arrow indicate the trajectory

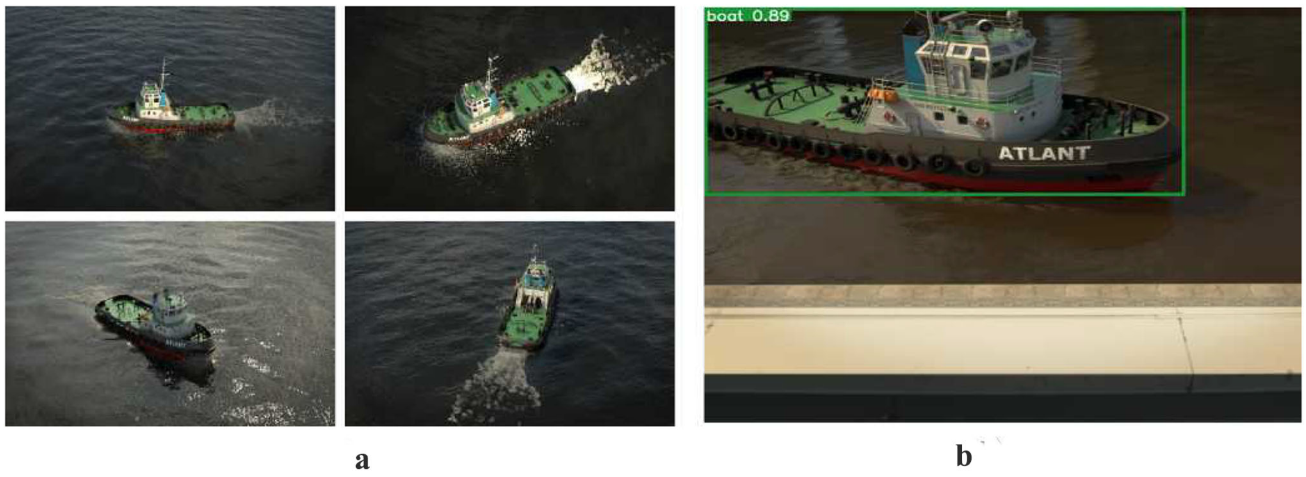

Fig. 3 Object detection examples. **a** Four samples of the additional rendered dataset for object detection training. **b** An example of tugboat detection using YOLOv5 [20] on one of the left images of the virtual stereo dataset. The detection is represented by the green bounding box and the number represents the detection confidence

4.2 Image segmentation

Given a valid object detection, this stage continues to compute the dense (meaning, pixel-wise) optical flow vectors of the current and previous left stereo frames. Optical flow is the amount of displacement between two frames and can be used to describe motion of the 2D image plane. To make the computation of a dense flow map possible in real time, this stage uses specialized video-encoding hardware integrated into the embedded device. More specifically, this is a hardware implementation of the pyramidal Lucas–Kanade optical flow [23] used for highly efficient video coding (HEVC) [24]. The Lucas–Kanade method works by minimizing the differences in image intensity over a local neighborhood using least-squares. The result is a pixel displacement vector that relates two points between the frames. The hardware implementation requires the image to be monochromatic and rescaled to a resolution of (1920×1080). Optical flow is then computed using a 4×4 granularity, so that the output image has a resolution of (480×270).
The flow magnitude is computed to create a binary image based on an empirical fixed threshold and then constrained

to the detected bounding box, yielding an instance mask. The threshold was estimated by maximizing the intersection over union between generated and ground-truth mask. This was done once in an offline regression for the whole system. A low threshold results in outliers from the water body while a very high threshold results in holes across the mask. This parameter needs to be further investigated in different scenarios to see if it is scene, object or speed dependent.

4.3 Depth estimation

Since the previous step can be offloaded to specialized hardware, the depth estimation stage can run in parallel to the image segmentation. Inference of depth from two stereo frames is key for 3D reconstruction as it allows re-projection of image points into a 3D point cloud. While several geometric techniques exist to compute depth from 2D features, this work uses semi-global matching (SGM) [25] to compute a dense depth map for every pixel. The method finds the disparity for every point between two stereo frames by minimizing both a local matching term and a wider (hence the name, semi-global) regularization term. Disparity is the horizontal distance between two pixels and can be converted to depth using the physical distance between two cameras and the focal lengths, respectively. SGM is linearly independent for every pixel and can be efficiently computed on the GPU in parallel. Since the depth error grows quadratically with this method and the disparity is related to the image resolution, only a limited field of view is supported. This work currently aims at harbor basins and water locks where a physical range from ∼25 to ∼65 m is typical.

4.4 Object tracking

In order to integrate frame sequence over time into a consistent 3D representation, we need to track the object's motion. The centroid of the first detected position is assumed to be origin of a local object coordinate system. Currently only one object at a time is considered. However, when extending this formulation to multiple objects, each object will have its own local coordinate system. All subsequent motion of the detected object is relative to that reference frame. Since the integration requires the voxel volume to be aligned with the object's position, we perform pose estimation for six degrees of freedom using the iterative-closest point (ICP) algorithm [26]. By masking out all static depth information, we can treat the dynamic object as static and perform camera tracking. The camera motion can be expressed as a matrix. The inverse of that matrix keeps the camera fixed and thus describes the objects motion (so-called ego-motion). We use the metric summarized by [27] combined with parallel ICP implementations presented by [6,8]. However, we use an optimized implementation presented by [2] that takes advantage of partial matrix multiplications available on the GPU device. As proposed by [6], we use a frame-to-model approach using ray casting to improve the robustness of the system.
Furthermore, we include a distance constraint to the tracking system that is illustrated in Fig. 4. The figure shows the ground-truth trajectory as seen from the camera in green (lower curve). The upper curve (blue) is the projected centroid as seen from the camera. This is used to constrain motion. The associated points (dashed, zigzag lines) show the correspondence between ground-truth and estimated positions.

We re-project the centroid of a masked depth map into the camera frame to constrain object motion. Centroids are smoothed using a simple running average. By limiting the difference to the distance between the current and previous observed centroids, we avoid sudden failure of the tracking system. To compute the centroids, we project all depth pixels inside the generated mask (valid pixels) to 3D space and compute the mean for the X- and Y-axis using GPU acceleration. For Z, a mean of zero is assumed. The difference in centroids projected onto the XY-plane between a current frame f_t and a frame at f_{t-1} constraints the translation. A downside of this algorithm is that partial occlusion or shifts in shape also lead to shifts in the centroid, henceforth weakening the constraint. This is shown in Fig. 4 when examining the distribution of centroid estimates.

4.5 Volumetric integration

Estimated depth, object tracking information and the left camera color frame are then fused into a consistent volumetric display using projective truncated signed distance fields (TSDF). For this, we developed our own implementation of [6], adapted to our target system. This was done because the original work from [6] is not publicly available and the implementation by [28] is only suited for static scenes. The general approach, however, is well suited for the reconstruction of dynamic objects as shown in [7,11]. For better color display, we included a projection-based filter and constrained the truncation distance to a narrower band around the iso-surface to avoid mismatches due to the overall scale of maritime environments. An important detail is the use of an adaptive weighting scheme. Each time a voxel is successfully updated, its weight value is increased by one. At the same time, a global weight counter is increased every time a volume integration happens. These two values are used in the next stage for filtering. And finally, we currently limit the maximum resolution to 10 cm for depth error compensation. This is related to the properties of the TSDF volume: Since every update to the voxel volume is weighted, the TSDF acts as a running average filter over time. However, if the voxel size is larger than the expected error of the observed depth, the TSDF cannot converge to the true iso-surface as the vox-

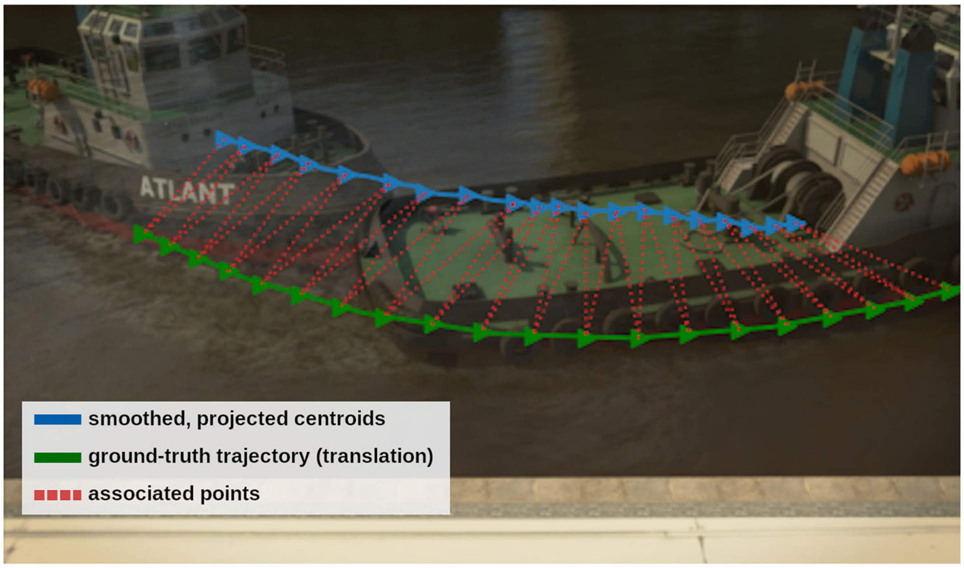
ATLANT
smoothed, projected centroids
ground-truth trajectory (translation)
associated points

Fig. 4 An illustration of the difference between centroid and ground-truth translation of the vessel. For visualization both trajectories were projected into the camera frame. First and last image from the dataset are overlayed for clarity

els enclose the error. Therefore, we set the resolution to a smaller value that captures not only the observed value but also allows for an uncertainty region to improve convergence behavior. For more details, refer to the original paper by [6]. Special care must be taken when considering the size of the volume since the memory requirements grow with $\mathcal{O}(N^3)$ where N is the number of voxels along one dimension.

4.6 Point cloud generation

To provide a suitable model for 3D display for end users, we extract and filter the voxel volume to gain a dense point cloud. This step happens only on demand and not every frame. Besides 3D positions, the point cloud also contains information about the RGB color and the geometric surface normal vector of every point as well as the distance to the estimated iso-surface (given by the TSDF value). The surface normal is calculated by evaluating the TSDF values around each point as outlined in [6]. Our novel multistage filtering performs voxel garbage collection similar to [4,9] but improving on the speed and overall methods of filtering.
First, we propagate through the voxel volume to fill a label buffer with binary values by filtering each TSDF value. We test every voxel and assign 1 if the weight of this value is above a certain threshold N and 0 otherwise. This means that only voxels that have been updated at least N times are kept under the assumption that those are inliers. We found empirically that 10% is already enough to remove most of the noise. An example of this will be discussed in Sect. 5.6.
Next, we apply a band-pass filter to extract only a narrow area around the iso-surface. During integration, the possible range is clipped to $[-1, 1]$. Through testing, we found that outside of the range $[-0.9, 0.9]$ only noise is present. We keep only values within this band when extracting the points.

Using the label buffer containing binary values for every voxel, we run a GPU-accelerated conditional select using the Nvidia CUB library [29] that outputs a set of points with label 1. Lastly, a comb filter is used to filter out points based on their hue spectrum. Since water has a very distinct hue range, we found that ships have a good contrast to allow filtering. This filter can be used dynamically by the end user depending on the scene. When displaying the point-cloud the user can select a hue range that they wish to be removed and the rendering algorithm will dynamically skip points whose color falls within the hue range.

5 Results

Using the simulated dataset, we verified our pipeline. The overall runtime of our pipeline is ∼161.95 ms (∼6.2 Hz). For every stage, we generated the respective ground-truth data and compared it to the estimated outputs. In the following sections, we discuss the results of each stage.

5.1 Object detection

For this stage, we selected the lightest configuration of YOLOv5, named YOLOv5-Nano. It provides a compromise between memory footprint, inference time and precision. We start training with pre-trained weights on the MS COCO dataset [21]. We extended the model using a batch size of 8 and trained for 50 epochs using a single class *boat*. Input size was 640 × 640. As training data, we used an additional synthetic dataset containing multiple views of the tugboat that we created ourselves (refer to Fig. 3a). This configuration allows for an efficient inference within the embedded system. We obtain an average precision (AP) of 0.907 and

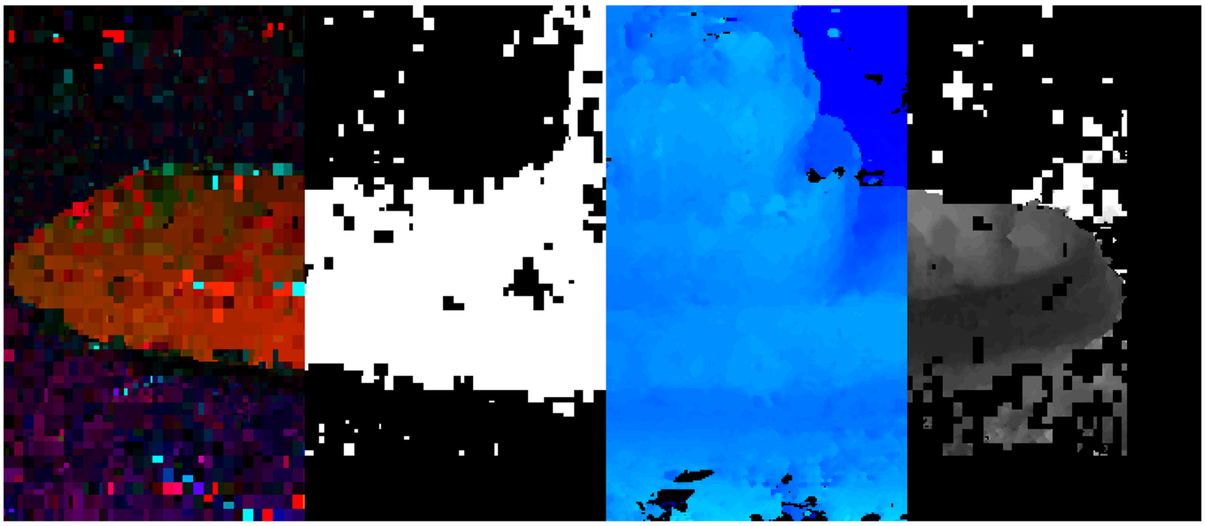

Fig. 5 Different outputs from successive pipeline stages. From left to right: Optical flow vectors (color encoded with flow as saturation and direction as hue), binary instance mask, estimated disparity, physical depth map with mask applied

average inference time on the Jetson AGX Xavier of 74.436 ms per frame.

Our AP is higher in comparison with a recent work by [30] focusing on instance segmentation of ships using Mask R-CNN [12] (0.772) and DetectoRS [31] (0.78). Since we regress the mask later in the pipeline using optical flow, we used a simpler and faster bounding box estimation. The increase in AP relates to the difference in complexity when comparing bounding box estimation with instance segmentation. The work performed by [32] shows a similar object detection pipeline as used in this work. The core difference is that [32] used YOLOv4 [33] while we used YOLOv5, allowing us to achieve a better AP than in their work (0.7586).

5.2 Image segmentation

Instance segmentation runs on average in 29.54 ms including optical flow estimation and mask creation. The AP is 0.68 showing a good overall match with the ground-truth mask. Figure 5 shows the optical flow vectors and the binary mask (from left). Experiments showed no significant benefit of using the average flow direction within the detected bounding box when computing the binary mask. This is due to the fact that direction of water flow and reflections on the water surface can be similar to the target motion. Since optical flow does not semantically differ between elements, the optical flow direction is ambiguous. Optical flow magnitude is dependent on the scene intensity (i.e., the lighting and surface of the observed target). To alleviate this, we tested adaptive thresholding based on mean and mode of observed flow magnitudes. Since those results were varying greatly from frame to frame, we opted for a fixed threshold of 1.7 to proof the general concept. This value, however, is scene (especially lighting) dependent. Hard shadows as encountered in direct sunlight create high-contrast regions that spread out the magnitude within the detected bounding box. Therefore, we continue to focus on adaptive thresholding to yield more stable results in a wide variety of scenes. A density-based clustering algorithm like DBSCAN [34] can help to dynamically create magnitude clusters that effectively reject outliers while respecting the overall shape of the optical flow. While performance and precision are good, the major disadvantage of this technique is currently the necessity of object motion. However, this work is concerned with dynamic reconstruction for situational awareness pictures. Therefore, non-moving objects are only of limited interest as other methods of observation exist.

5.3 Depth estimation

To support medium-scale environments, we chose a baseline of 4 m between cameras. We used a disparity of 256 pixels as a tradeoff between speed and image resolution. The maximum scene depth is constrained by the depth error which increases quadratically with depth. Limiting this error to the desired value sets the upper boundary for scene depth. The minimum scene depth is restricted by the maximum disparity. Objects closer to the camera will diverge more (i.e., the horizontal pixel difference increases). The dataset used in this work was set up to keep the harbor basin within those depth boundaries. While the minimum scene depth was set to $\sim$10 m, the tugboat we used in dataset only moves between $\sim$ 25 and $\sim$ 65 m depth. We plotted the current disparity and depth error as functions of depth in Fig. 6. The vertical orange bars refer to the movement range of the object of interest (tugboat). The curves start at the minimum scene depth (as outlined in

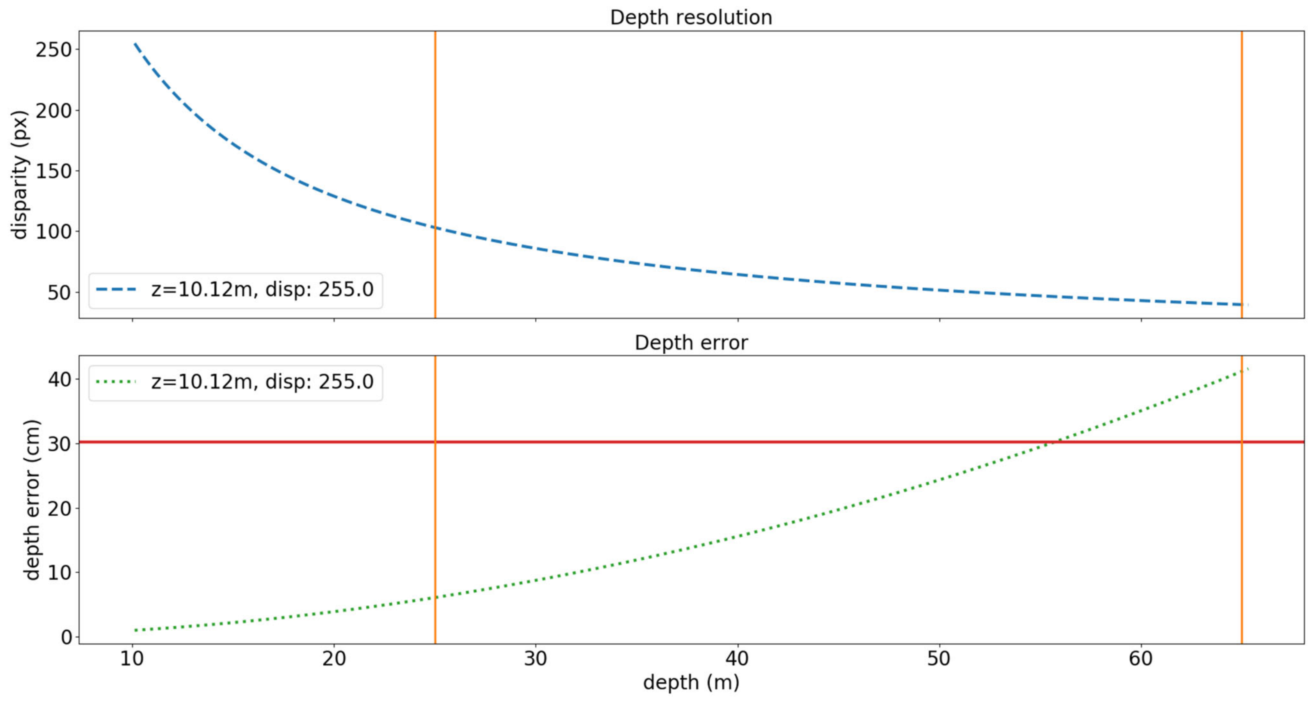

Fig. 6 Disparity (dashed blue line, top) and depth error (dotted green line, bottom) as functions of depth. The horizontal red line (bottom) marks the current mean average error for the evaluated scene setup. The vertical orange bars indice the set depth limits for the scene

the respective legends). Using this setup the mean absolute error (MAE) is ∼0.3022 m (horizontal red bar in Fig. 6) with estimation taking 24.54 ms on average. Considering ∼40 m depth of the test scene, the error is only 0.75% which is low. Given this setup, we want to show two ways of optimizing for maximum disparity or camera baseline, respectively.

1. Due to the left vertical orange line intersecting the disparity curve at a value of 102, the maximum disparity can be reduced to improve overall performance without sacrificing quality.
2. The minimum scene depth (10.21 m) can be shifted to the right until it reaches the minimum scene depth (25 m). This is achieved by extending the baseline between the two cameras. While this would significantly reduce the error, it also makes mounting the cameras more difficult in a physical environment.

These constraints regarding optimal depth estimation are currently scene dependent. They must be set up as a compromise between field of view and expected object size. In Sect. 7, we discuss a third alternative to the above-mentioned optimizations for long-range depth estimation. For a visual reference on the output of this pipeline stage, Fig. 5 shows the disparity image in the blue column (third from left).

5.4 Object tracking

Since the object tracking stage uses ray casting to fit the current depth map to the already integrated model, we provide two timings. The ICP method itself runs in ∼2.66 ms and the ray casting takes ∼27.67 ms. While the addition of ray casting significantly increases the processing time, it also stabilizes the integration stage.

The overall translational error of the object tracking stage is ∼8.64 m and the rotational errors for all three Euler angles α, β, γ for axis XYZ are $\alpha = 3.85$, $\beta = 5.31$, and $\gamma = 1.95$ degrees, respectively. Although the error is high compared to areas like autonomous robotics the scale here is also larger. With a maximum scene depth of ∼65 m, the error corresponds to 13.3%. This error stems from translational drift that occurs over time, meaning the offset between ground-truth and observed trajectory increases. While this error is currently large, our reconstruction system still manages to output an approximation of the observed object with distinct features and color. For geo-referencing and tracking of the object, our error is still lower than the 10 m generated through homography as presented by [30] for closer than 400 m to the observer.

5.5 Volumetric integration

Table 1 shows a comparison between different voxel resolutions for the integration stage. For comparison, we run the

Table 1 Results for volumetric integration stage

Voxel resolution (m)	Volume dim	Memory usage (MB)	Time (ms)
0.4	128^3	25.2	1.01
0.2	256^3	201.3	1.02
0.1	512^3	1610.6	1.05

pipeline with ground-truth inputs from the dataset to generate an optimal result for every resolution. Despite all voxel resolutions requiring almost equal update time, their memory usage differs significantly. This is an important consideration on an embedded system. The timing similarities can be explained by memory transfers and pipeline stalling in the GPU@. For single-object 3D reconstruction a volume of 512^3 is feasible for real-time processing with improved accuracy and details. For multiple objects, a resolution of 0.2 m and a volume dimension of 256^3 yield a good compromise between speed, details and memory consumption. While a volume dimension of 1024^3 was tested, it caused the system to run out of memory.

5.6 Point cloud generation

Figure 7 shows the output of the pipeline in comparison with a frame from the dataset. While certain details are obscured by noise, the overall shape, color and geometry are present. Even finer details like the stairs on the back of the bridge are seen in the reconstruction.

Figure 8 shows a visual comparison of the unfiltered pipeline output (a) and with filtering applied (b). The voxel garbage collection efficiently removes outlier noise while improving the overall shape with only high-confidence points being kept. Also, the density and file size of the point-cloud, was reduced from $2,538,535$ points (337 MB uncompressed) to $516,464$ points (68 MB uncompressed). Thus, the garbage removal yields a $\sim$4.9 reduction factor while improving readability.

6 Implications

In this section, the implications for the maritime field will be discussed. The framework presented here aims to provide a foundation to replace traditional 2D situational awareness pictures. While the generation of static 3D maps and high-resolution scans of outdoor environments can be considered robust and readily available, a key aspect of 3D situational awareness pictures is dynamic content. This work provides a working framework to generate the required data to populate 3D situational awareness pictures with real-time information about dynamic objects. It has operational implications for how situational awareness is communicated and assessed by stakeholders, may it be law enforcement or port authorities. Thus, a new approach to the operation of maritime safety and security evaluation is provided.

As the research shown here presents only a first proof of concept, the next step will evaluate how integration of a dynamically 3D reconstructed object into a 3D situational awareness picture can be performed. This could provide operators with a completely new way of interacting with situational awareness pictures. A full 3D display consisting of high-resolution static maps and dynamic objects updating in real time. Since all content is communicated in 3D, the foundation for new technologies like mixed reality is given. Interaction with situational awareness pictures can be more immersive and semantically clearer while collaboration of different operators is simplified when evaluating the security status of maritime infrastructures. Additionally, the framework presented here is perceived as a tool to simplify techniques such as 3D anomaly detection, autonomous navigation and collision avoidance in harbor basins and waterways.

To enhance safety and security in the maritime field, the proposed work could act as a key technology that enables unified monitoring of large ports areas. Multiple instances of the proposed work generating 3D representations of dynamic objects in a port could provide end users with a dense representation of their surrounding that would not have been possible with traditional sensing alone. With the current work as a proof of concept, research needs to focus on outdoor lighting and weather. This is especially important since weather effects (fog, rain, sun glare) can greatly effect the perception of objects in a maritime environment. Harsh shadows from direct sunlight and diffusion due to mist can also impact the readability of an object and henceforth the errors associated with 3D reconstruction. For robust outdoor use, the system needs to be designed around such effects. Also, since maritime traffic operates around the clock, the system eventually needs to be extended by a formulation for nighttime observation. This includes research assessing the feasibility of laser sensing and (near) infrared cameras. Therefore, this work will be further researched to determine the feasibility of above-mentioned ideas and verify the operational implications.

7 Conclusion and future work

This work presents a novel framework to generate 3D models from stationary camera systems for improving monitoring in maritime infrastructures and for integration into situational awareness systems. We describe and evaluate a dynamic 3D reconstruction pipeline prototype that delivers valid results for single maritime object detection and reconstruction. The

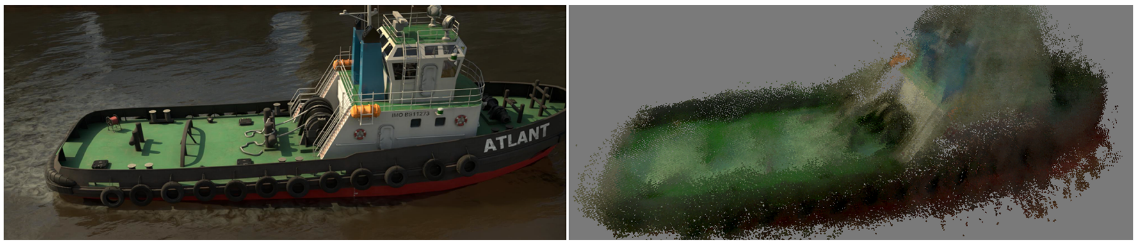

Fig. 7 Left: Left frame taken from the stereo dataset, input for our pipeline. Right: 3D reconstructed tugboat using our presented pipeline prototype for dynamic 3D reconstruction on an embedded system. A volume dimension of 512^3 with a voxel resolution of 0.1 m was used

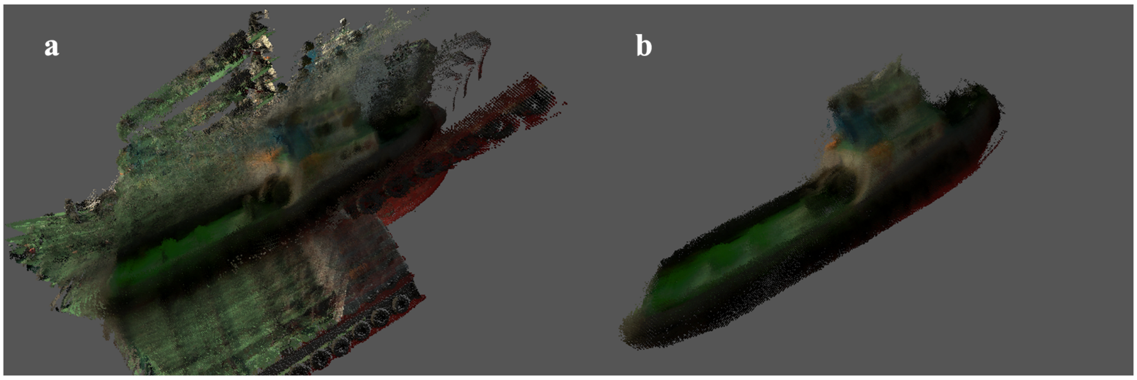

Fig. 8 3D reconstructed model with a volume dimension of 512 and a voxel resolution of 0.2 m. In **a**, voxel garbage collection is disabled, and in **b**, it is enabled. The garbage collection removes Ghosting artifacts and projection noise resulting in a well-defined silhouette. The confidence threshold for inliers was set to 10%, meaning only samples with a higher integration weight are taken into account. Clearly visible is the removal of outlier noise and the well-defined silhouette

system runs in near real time with ∼6 Hz on an GPU-accelerated embedded device. The system output is ready for import into a high-resolution 3D static map to enhance static situational awareness displays with consistent dynamic 3D objects. However, integrating the reconstructed object into the 3D static map requires the development of an automated pipeline. Maritime infrastructures are currently monitored with the help of stationary video cameras. Observing a large area in detail requires human operators to investigate several video feeds. Each video feed displays only its own field of view (referred to as context). Our approach reduces the need to switch between multiple video feeds (context-switching) and instead aims to create a more unified approach to situational awareness.

While the overall pipeline provides valid results, the pose prediction and instance segmentation require the object to be in motion. As a next step we want to refine these stages to support also semi-static objects that only move sporadically. Furthermore, instance segmentation can be refined using adaptive thresholding as outlined in Sect. 4.2. An adaptive threshold algorithm would be especially beneficial when multiple objects of different speeds or under varying lighting conditions are observed. Recent work shown by [35] also demonstrates that the tasks of object detection, instance segmentation and pose estimation can be jointly estimated using a single neural network. With embedded systems becoming more powerful, computationally efficient variants of more demanding networks can be run in real time. Therefore, we aim to unify stages outputting bounding box, mask and pose in a single step without sacrificing the real-time constraint.

To improve large-scale depth estimation, an adaptive framework can be used. By limiting the depth estimation to areas around detected bounding boxes, larger image resolutions become feasible without penalizing performance. Higher-resolution images offer more features per object and thus increase the range over which a stereo system can operate. To further reduce the depth error over large scene depths, statistical shape priors [3] and probabilistic fusion techniques [11] can be used.

Acknowledgements An earlier version of this work was published as part of the Proceedings of the second European Workshop on Maritime Systems Resilience and Security (MARESEC 2022) under the title “Real-time embedded reconstruction of dynamic objects for a 3D maritime situational awareness picture”.

Funding Open Access funding enabled and organized by Projekt DEAL. This research was funded by the core funding of the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt e.V.). The funding is granted by the Federal Ministry for Economic Affairs and Climate Action (Bundesministerium für Wissenschaft und

Klimaschutz, BMWK). It has been realized as part of the project "Digital Atlas" in a cooperation between the DLR institute for the protection of maritime infrastructures and the DLR institute for optical sensor systems.

Data availibility This work uses an internal dataset generated by the German Aerospace Center (Deutsches Zentrum für Luft- und Raumfahrt e.V.) that contains non-public information and is not disclosed to the public.

Declarations

Conflict of interest This work is not attached to any other third-party stakeholders and does not imply any conflict of interests.

References

1. Sirimanne, S.N.: Review of Maritime Transport 2020. United Nations Publications, New York (2020)
2. Whelan, T., Kaess, M., Johannsson, H., Fallon, M.F., Leonard, J.J., McDonald, J.B.: Real-time large scale dense RGB-D SLAM with volumetric fusion. Int. J. Robotics Res. IJRR (2014). https://doi.org/10.1177/0278364914551008
3. Engelmann, F., Stuckler, J., Leibe, B.: SAMP: Shape and motion priors for 4d vehicle reconstruction. In: 2017 IEEE Winter Conference on Applications of Computer Vision (WACV) (2017). https://doi.org/10.1109/wacv.2017.51
4. Barsan, I.A., Liu, P., Pollefeys, M., Geiger, A.: Robust dense mapping for large-scale dynamic environments. In: 2018 IEEE International Conference on Robotics and Automation (ICRA) (2018). https://doi.org/10.1109/icra.2018.8462974
5. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the Kitti dataset. Int. J. Robot. Res. **32**(11), 1231–1237 (2013). https://doi.org/10.1177/0278364913491297
6. Newcombe, R.A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A.J., Kohi, P., Shotton, J., Hodges, S., Fitzgibbon, A.: Kinectfusion: Real-time dense surface mapping and tracking. In: 2011 10th IEEE International Symposium on Mixed and Augmented Reality, pp. 127–136 (2011). https://doi.org/10.1109/ISMAR.2011.6092378
7. Grinvald, M., Tombari, F., Siegwart, R., Nieto, J.: TSDF++: a multi-object formulation for dynamic object tracking and reconstruction. In: 2021 IEEE International Conference on Robotics and Automation (ICRA), pp. 14192–14198 (2021). https://doi.org/10.1109/ICRA48506.2021.9560923
8. Curless, B., Levoy, M.: A volumetric method for building complex models from range images. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques. SIGGRAPH '96, pp. 303–312. Association for Computing Machinery, New York (1996). https://doi.org/10.1145/237170.237269
9. Nießner, M., Zollhöfer, M., Izadi, S., Stamminger, M.: Real-time 3D reconstruction at scale using voxel hashing. ACM Trans. Graph. **32**(6) (2013). https://doi.org/10.1145/2508363.2508374
10. Bullinger, S.: Image-based 3D reconstruction of dynamic objects using instance-aware multibody structure from motion. Ph.D. thesis, Karlsruher Institut für Technologie (KIT) (2020). https://doi.org/10.5445/KSP/1000105589
11. Strecke, M., Stückler, J.: EM-fusion: Dynamic object-level SLAM with probabilistic data association. In: Proceedings IEEE/CVF International Conference on Computer Vision 2019 (ICCV), pp. 5864–5873. IEEE (2019). https://doi.org/10.1109/ICCV.2019.00596
12. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: 2017 IEEE International Conference on Computer Vision (ICCV), pp. 2980–2988 (2017). https://doi.org/10.1109/ICCV.2017.322
13. Zhang, H., Liu, Z.-Q., Wang, Y.-L.: U-loam: A real-time 3D lidar SLAM system for water-surface scene applications. In: 2022 IEEE International Conference on Unmanned Systems (ICUS), pp. 653–657 (2022). https://doi.org/10.1109/ICUS55513.2022.9986766
14. Blender Foundation: Blender: a 3D modelling and rendering package, Stichting Blender Foundation, Amsterdam (2018). http://www.blender.org
15. Ribeiro, M., Damas, B., Bernardino, A.: Real-time ship segmentation in maritime surveillance videos using automatically annotated synthetic datasets. Sensors **22**, 8090 (2022). https://doi.org/10.3390/s22218090
16. Denninger, M., Sundermeyer, M., Winkelbauer, D., Olefir, D., Hodan, T., Zidan, Y., Elbadrawy, M., Knauer, M., Katam, H., Lodhi, A.: Blenderproc: Reducing the reality gap with photorealistic rendering. In: Robotics: Science and Systems (RSS) (2020). https://elib.dlr.de/139317/
17. Károly, A.I., Galambos, P.: Automated dataset generation with blender for deep learning-based object segmentation. In: 2022 IEEE 20th Jubilee World Symposium on Applied Machine Intelligence and Informatics (SAMI), pp. 000329–000334 (2022). https://doi.org/10.1109/SAMI54271.2022.9780790
18. Qiu, W., Yuille, A.: Unrealcv: connecting computer vision to unreal engine. In: Hua, G., Jégou, H. (eds.) Computer Vision - ECCV 2016 Workshops, pp. 909–916. Springer, Cham (2016)
19. Yang, Y., Lu, C.: A stereo matching method for 3d image measurement of long-distance sea surface. J. Mar. Sci. Eng. **9**(11) (2021). https://doi.org/10.3390/jmse9111281
20. Jocher, G., Chaurasia, A., Stoken, A., Borovec, J., Kwon, Y.: ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference. Zenodo (2022). https://doi.org/10.5281/zenodo.6222936
21. Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) Computer Vision – ECCV 2014, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
22. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d' Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019). http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
23. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume

2. IJCAI'81, pp. 674–679. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1981)
24. Sullivan, G.J., Ohm, J.-R., Han, W.-J., Wiegand, T.: Overview of the high efficiency video coding (HEVC) standard. IEEE Trans. Circuits Syst. Video Technol. **22**(12), 1649–1668 (2012). https://doi.org/10.1109/TCSVT.2012.2221191
25. Hirschmüller, H.: Stereo processing by semiglobal matching and mutual information. IEEE Trans. Pattern Anal. Mach. Intell. **30**(2), 328–341 (2008). https://doi.org/10.1109/TPAMI.2007.1166
26. Rusinkiewicz, S., Levoy, M.: Efficient variants of the ICP algorithm. In: Proceedings Third International Conference on 3-D Digital Imaging and Modeling, pp. 145–152 (2001). https://doi.org/10.1109/IM.2001.924423
27. Chen, Y., Medioni, G.: Object modeling by registration of multiple range images. In: Proceedings. 1991 IEEE International Conference on Robotics and Automation (ICRA), pp. 2724–27293 (1991). https://doi.org/10.1109/ROBOT.1991.132043
28. Diller, C.: KinectFusionLib (2018). https://github.com/chrdiller/KinectFusionLib
29. Merrill, D.: CUB v1.16.0. Nvidia Research (2022). https://nvlabs.github.io/cub/index.html
30. Carrillo-Perez, B., Barnes, S., Stephan, M.: Ship segmentation and georeferencing from static oblique view images. Sensors **22**(7) (2022). https://doi.org/10.3390/s22072713
31. Qiao, S., Chen, L.-C., Yuille, A.: Detectors: detecting objects with recursive feature pyramid and switchable atrous convolution. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10213–10224 (2021)
32. Solano-Carrillo, E., Carrillo-Perez, B., Flenker, T., Steiniger, Y., Stoppe, J.: Detection and geovisualization of abnormal vessel behavior from video. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), pp. 2193–2199 (2021). https://doi.org/10.1109/ITSC48978.2021.9564675
33. Wang, C.-Y., Bochkovskiy, A., Liao, H.-Y.M.: Scaled-yolov4: Scaling cross stage partial network. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 13024–13033 (2021). https://doi.org/10.1109/CVPR46437.2021.01283
34. Ester, M., Kriegel, H.-P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. KDD'96, pp. 226–231 (1996). https://doi.org/10.5555/3001460.3001507
35. Peng, S., Liu, Y., Huang, Q., Zhou, X., Bao, H.: Pvnet: pixel-wise voting network for 6dof pose estimation. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4556–4565. IEEE Computer Society, Los Alamitos, CA, USA (2019). https://doi.org/10.1109/CVPR.2019.00469

Felix Sattler has been a scientific researcher since 2021 at the German Aerospace Center (DLR) Institute for the Protection of Maritime Infrastructures with a background in computer graphics and image processing. His research focuses on 3D reconstruction of dynamic scenes and real-time high-resolution image processing for embedded systems.

Borja Carrillo-Perez has been a scientific researcher since 2018 at the German Aerospace Center (DLR) Institute for the Protection of Maritime infrastructures. His research activities include the development of image processing instruments, research and application of deep learning algorithms, real-time data analysis and patter recognition for the improvement of safety and security in maritime infrastructures.

Sarah Barnes is the head of the Methods and Processing group at the DLR Institute for the Protection of Maritime Infrastructures. Sarah received her PhD from the University of Manchester in 2016 in the field of experimental particle physics. Her broad research interests now lie in the fields of cosmic-ray tomography, optical sensor systems, sonar and machine learning.

Karsten Stebner is a researcher at the Institute of Optical Sensor Systems at German Aerospace Center (DLR). His research interests are in the area of geo-referencing of complex configurations and systems, dense image matching and surface reconstruction.

Maurice Stephan is head of the Department of Maritime Security Technologies at the Institute for the Protection of Maritime Infrastructures which is part of the German Aerospace Center (DLR). He received his PhD from RWTH Aachen University, Germany, where he conducted research in experimental astroparticle physics. His research interest focuses on instrumentation for security applications and related data processing and analysis mainly for optical and hydroacoustic instruments as well as scanning systems based on muon tomography. In addition, his research interest includes heterogenous multi-sensor networks for maritime situational awareness systems.

Gregor Lux is a professor for computer graphics at the Westphalian University of Applied Sciences, Gelsenkirchen, Germany. He received his doctorate (Dr.-Ing.) from the Technical University in Darmstadt, Germany in 1991. From 2007 to 2016, he was the dean of the "computer science and communication departmen" at Westphalian University. His research interests include tracking technologies and spacial interaction design for virtual environments, XR-based education and XR games.