Real-time embedded reconstruction of dynamic objects for a 3D maritime situational awareness picture

Felix Sattler*, Sarah Barnes*, Borja Carrillo-Perez*, Karsten Stebner†, Maurice Stephan* and Gregor Lux‡
*Institute for the Protection of Maritime Infrastructures, German Aerospace Center (DLR), Bremerhaven, Germany
Email: {felix.sattler, sarah.barnes, borja.carrilloperez, maurice.stephan}@dlr.de
†Institute of Optical Sensor Systems, German Aerospace Center (DLR), Berlin, Germany
Email: karsten.stebner@dlr.de
‡Westphalian University of Applied Sciences, Gelsenkirchen, Germany
Email: gregor.lux@w-hs.de

***Abstract*—Monitoring maritime infrastructures is an important part of maritime safety and security. To best assess the security status of these facilities, detailed information should be made available to stakeholders, such as port authorities, law enforcement agencies and emergency services in a concise and easily understandable format. In this work, we propose a novel real-time 3D reconstruction framework for enhancing maritime situational awareness. 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. Our pipeline runs with ∼6Hz on a Nvidia Jetson Xavier AGX embedded system and is verified using a simulated dataset of a harbor basin.**

***Index Terms*—situtational awareness, dynamic 3D reconstruction, real time monitoring, maritime safety and security**

I. Introduction

The maritime environment is home to many driving factors of modern society. Maritime transport, for example, is a crucial element of global trading, helping to facilitate a close interdependency between countries, manufacturers and markets [1]. Maritime infrastructures such as harbour areas and cargo terminals, are critical for the sucessful 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. 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. While in the domains of autonomous driving and robotics, simultaneous localisation and mapping (SLAM) systems that employ 3D reconstruction are the main focus of research, our approach is fundamentally different.

A vast body of research exists for indoor, small-scale RGB-D[1] systems in the context of autonomous robotics [7] as well as on LiDAR[2] and stereoscopic outdoor 3D reconstruction for autonomous driving [21] [8]. Due to the sensor setups for moving observers as shown in [2] a specific motion model is implicitly assumed. In our use case, which deals with static camera views, no moving observer is present which limits the use of these existing works. While LiDAR technology is also researched in great detail, the lasers struggle with water surfaces as shown in [3] leading to distortions and the necessity of extensive filtering.

This work therefore presents a novel framework for enhancing maritime situational awareness by introducing a pipeline prototype for dynamic 3D reconstruction of maritime objects using a static observer and stereoscopic cameras on an GPU-accelerated embedded device. As a proof of concept, the pipeline presented in this work was tested and verified using a 3D simulation in a virtual, geo-referenced environment for a single dynamic object only.

II. Related work

Dynamic 3D reconstruction can be split into different subproblems with their respective research bodies associated with them. Therefore, this section will focus on complete, state-of-the-art 3D reconstruction systems only.

[4] 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 (∼2Hz). [5] proposes improvements to this work through the incorporation of auxiliary

[1]RGB-D: RGB camera with short-range, active depth sensors
[2]LiDAR: Light detection and ranging

data like multiple objects and color data. A core aspect of [4] is the use of truncated signed distance fields (TSDF) and a volumetric grid to fuse depth information into a consistent, implicit representation. Since TSDFs can be generated through projective data assocation [6], they can be efficiently computed in parallel and are suitable for a GPU device.

[7] adopted the framework by [4] 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 can not be used in outdoor environments with strong infrared lighting.

Therefore, a different system is required to infer good depth estimates. This was addressed by [8] who proposed a complete stereo-based hybrid real-time 3D reconstruction system. Developed in the context of autonomous driving, their system achieves 5Hz 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 [8] 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. [8], 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[3] network. 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 4-9Hz showing the high performance demand. The largest performance drop stems from the use of Mask R-CNN.

Our pipeline was developed as a modular multi-stage system like [8], theoretically supporting multi-object reconstruction similar to [5] and [11] with a unique volume per object. Instead of using deep learning for instance segmentation as [10] and [11], we use a simple 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 [8] and [9] are extended, implemented and verified in our presented pipeline.

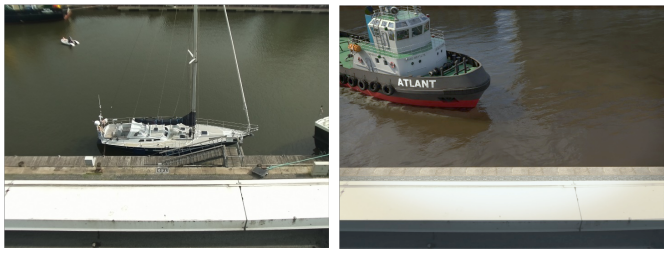

Fig. 1. A comparison between the real harbor basin (left) and the digital re-creation (right). Notable are the accuracy of reflections on the water surface, materials and general perspective.

III. Dataset creation

To validate our pipeline we created a virtual dataset in the open-source 3D software Blender3D [12]. 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. Depth is between $\sim$25m and $\sim$60m, motivated by typical observation scenarios at port basins and locks. To verify our dynamic 3D reconstruction system, we used a virtual re-creation of a real tugboat to scale ($\sim$35m length). The sequence is 25s long (624 frames). It was created as a stereoscopic dataset with a distance of four meters between cameras. Besides stereoscopic frames, the dataset contains ground-truth values for every pipeline stage including 3D shape, object transformation, camera geo-reference (UTM projection), depth information (1cm resolution), mask information and a bounding box.

IV. Dynamic embedded 3D reconstruction

Real-time dynamic 3D reconstruction is performed on a GPU-accelerated embedded device, the Nvidia Jetson Xavier

[3]Region-based convolutional neural network

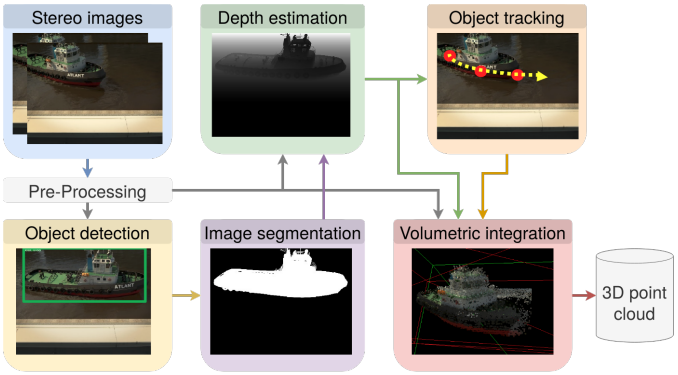

Fig. 2. A schematic view of the reconstruction pipeline with ground-truth examples for each stage. Images are captured by a stereoscopic camera. The object is detected on the left frame using deep learning. Following that, image segmentation is performed and combined with the bounding box to produce a mask of the detected dynamic object. Next, object depth is estimated with respect to the left frame. The resulting depth data is masked using the binary mask and fed to object tracking, resulting in a transformation matrix. Finally, the left input frame, depth and transformation data are fused into a volume to generate a 3D point cloud for situational awareness display.

AGX. Processing is done by a pipeline comprised of several successive stages. All computations are done on a per-frame basis with exception of the last stage, which is point cloud extraction. This is only done on demand when a new representation is requested. Figure 2 shows an overview of the processing pipeline, from left to right:

1) Stereoscopic camera frames are pre-processed to prepare a pipeline run. This includes scaling, color channel conversion as well as uploading the frames to the GPU device. The stereo frames are uploaded once, with all major processing done on the GPU.
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 pre-processing stage uses common computer vision techniques and is therefore not discussed.

A. 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 [13]. It is a state-of-the-art algorithm for real-time object detection trained on the MS COCO dataset [14] that can perform efficiently on an embedded system. The algorithm was deployed on embedded device using the Pytorch framework [15]. For a robust ship detection model, we rendered an additional training set of 114 images of a tugboat in different sizes and perspectives. In Figure 3 some samples of the additional training set and an example of the tugboat detection on our virtual stereo dataset (see III) are shown. If no object was detected during this stage, the pipeline starts over again.

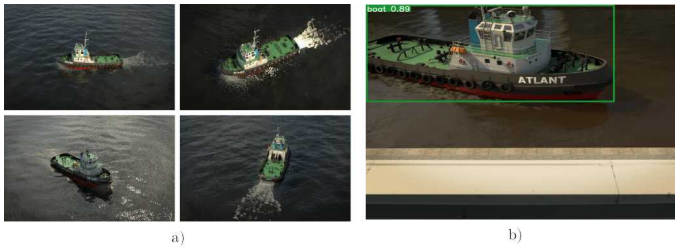

Fig. 3. Object detection examples. a) shows the four samples of the additional rendered dataset for object detection training. b) shows an example of tugboat detection using YOLOv5 [13] on one of the left images of the virtual stereo dataset. The detection is represented by the green bounding box and the number represents the detection confidence.

B. 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 [16]. 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 flow magnitude is then computed to create a binary image based on a fixed threshold. This is then constrained to the detected bounding box, yielding an instance mask.

C. 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) [17] 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 linear-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 $\sim$25 to $\sim$65 meters is typical.

D. Object tracking

In order to integrate several views (frames) into a consistent display, 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. All subsequent motion of the detected object is relative to that point. 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 [18]. 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 [19] combined with parallel ICP implementations presented by [6] and [4]. However, we use a optimized implementation presented by [7] that takes advantage of partial matrix multiplications available on the GPU device. As proposed by [4], we use a frame-to-model approach using ray-casting to improve the robustness of the system. Furthermore, we propose a novel constraint to the tracking system. We re-project the centroid of a masked depth map into the camera frame to constraint object motion. By limiting the difference to the distance between the current and previous observed centroids, we avoid sudden failure of the tracking system.

E. 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 [4], adapted to our target system. The approach is well suited for the reconstruction of dynamic objects as shown in [5], [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 12 cm for depth error compensation.

F. Point cloud generation

To provide a suitable model for display in a situational awareness map, 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 stores color and normal information as well as the distance to the estimated iso-surface (given by the TSDF value). Our novel multi-stage filtering performs voxel garbage collection similar to [9] and [8] 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. First, we apply a band-pass filter to extract only a narrow area around the iso surface. Then we reject all points if their integration weight is below a certain percentage of the global weight counter. 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. We reduce the voxel volume using the pre-filtered label buffer in parallel on the GPU to yield valid points.

V. Results

Using the simulated dataset, we verified our pipeline. The overall runtime of our pipeline is $\sim$ 161.95ms ($\sim$6.2Hz). Our dynamic 3D reconstruction system was tested on a Nvidia Jetson Xavier AGX embedded device with GPU-acceleration. 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.

A. Object detection

For this stage, we selected the lightest configuration of YOLOv5, named YOLOv5-Nano. It is a good compromise between memory footprint, inference time and precision. We start training with pre-trained weights on the MS COCO dataset [14] and extend the training for 50 more epochs with batch size eight with our additional rendered dataset containing multiple views of the tugboat. The image input size is 640×640px. This configuration allows for an efficient inference within the embedded system. We obtain an average precision (AP) of 0.907 and average inference time on the Jetson AGX Xavier of 74.436 ms per frame.

B. Image segmentation

Instance segmentation runs on average in 29.54ms 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). While performance and precision are good, the major disadvantage of this technique is currently the necessity of object motion.

C. Depth estimation

To support medium-scale environments, we chose a baseline of 4m and a maximum disparity of 256 pixels. The mean absolute error (MAE) is $\sim$ 0.3022m with estimation taking

Fig. 4. 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 reoslution of 0.1m was used.

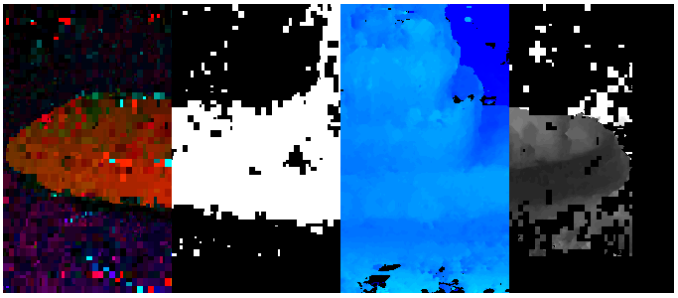

Fig. 5. Different outputs from successive pipeline stages. From left to right: Optical flow vectors (color encoded with flow as saturation and direction as hue), binary instance mask, estimated disparity, physical depth map with mask applied.

24.54ms on average. Figure 5 show estimated disparity (in blue) and the masked depth (far right) with details in bow and bridge being clearly distinguishable. Considering $\sim$ 40m depth of the test scene, this is only 0.75% which is low. The results are scene-dependent. Therefore, camera baseline and maximum disparity of the stereo setup must be optimized for optimal overall performance without sacrificing quality.

D. 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 $\sim$ 2.66ms and the ray-casting takes $\sim$ 27.67ms. While the addition of raycasting significantly increases the processing time, it also stabilizes the integration stage. The overall translational error of the object tracking stage is $\sim$ 8.64m 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 $\sim$ 65m, the error corresponds to 13.3%. For geo-referencing and tracking the object this provides an acceptable error. Also, since we are more interested in a complete reconstruction than a precise localisation, we can neglect drifting.

E. Volumetric integration

Table I shows a comparison between different voxel resolutions for the integration stage. For comparison we run the 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 0.2m 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.

TABLE I
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

F. Point cloud generation

Figure 4 shows the output of the pipeline in comparison to a frame from the dataset. While certain details are occluded by noise, the overall shape, color and geometry is present. Even finer details like the stairs on the back of the bridge are seen in the reconstruction. Figure 6 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 (337MB uncompressed) to $516,464$ points (68MB uncompressed). Thus, the garbage removal yields a $\sim$ 4.9 reduction factor while improving readability.

VI. CONCLUSION

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

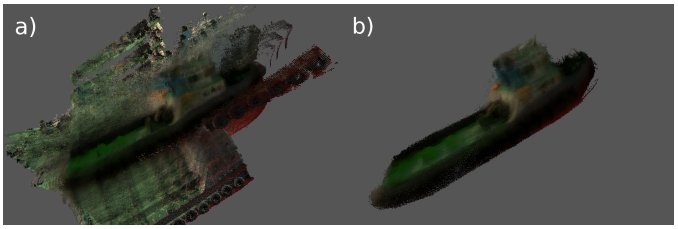

Fig. 6. a) The 3D reconstructed model with a volume dimension of 512^3 and a voxel resolution of 0.2m. Voxel garbage collection is disabled. Ghosting artifacts and projection noise are clearly visible. b) All parameters are identical to a) but voxel garbage collection is enabled. The confidence threshold 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.

awareness systems. We describe and evaluate a dynamic 3D reconstruction pipeline prototype that delivers valid results for single maritime object detection and reconstruction. The system runs in near real-time with $\sim$ 7Hz 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. Our approach reduces the need for context-switching between multiple video feeds and object displays. It aims to create a more unified approach to situational awareness.

VII. Future Work

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. The tasks of object detection, instance segmentation and pose estimation can be trained using a single back-end, as shown by [20]. Therefore, we aim to create a unified stage that outputs bounding box, mask and pose in a single step without sacrificing the real-time constraint. Furthermore, we plan to refine the depth error by the use of statistical shape priors [21] and probabilistic fusion techniques [11] to support larger scene depths. Lastly, we aim to provide an automatic pipeline that integrates the reconstructed object into the high-resolution 3D static map.

References

[1] S. N. Sirimanne, *Review of Maritime Transport 2020*. New York, NY, US: United Nations Publications, 2020.

[2] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.

[3] G. Toscano, U. Gopalam, and V. Devarajan, "Auto hydro break line generation using lidar elevation and intensity data," in *Proceedings, ASPRS Conference, Louisville, Kentucky*, 2014.

[4] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohi, J. Shotton, S. Hodges, and A. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking," in *2011 10th IEEE International Symposium on Mixed and Augmented Reality*, 2011, pp. 127–136.

[5] M. Grinvald, F. Tombari, R. Siegwart, and J. Nieto, "TSDF++: A Multi-Object Formulation for Dynamic Object Tracking and Reconstruction," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 14 192–14 198.

[6] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '96. New York, NY, USA: Association for Computing Machinery, 1996, p. 303–312. [Online]. Available: https://doi.org/10.1145/237170.237269

[7] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. Leonard, and J. McDonald, "Real-time large scale dense RGB-D SLAM with volumetric fusion," *Intl. J. of Robotics Research, IJRR*, 2014.

[8] I. A. Bârsan, P. Liu, M. Pollefeys, and A. Geiger, "Robust dense mapping for large-scale dynamic environments," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2018*, May 2018.

[9] M. Nießner, M. Zollhöfer, S. Izadi, and M. Stamminger, "Real-time 3d reconstruction at scale using voxel hashing," *ACM Transactions on Graphics (TOG)*, 2013.

[10] S. Bullinger, "Image-based 3d reconstruction of dynamic objects using instance-aware multibody structure from motion," Ph.D. dissertation, Karlsruher Institut für Technologie (KIT), 2020.

[11] M. Strecke and J. Stückler, "EM-fusion: Dynamic object-level slam with probabilistic data association," in *Proceedings IEEE/CVF International Conference on Computer Vision 2019 (ICCV)*. IEEE, Oct. 2019, pp. 5864–5873.

[12] B. Foundation, "Blender - a 3d modelling and rendering package," Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. [Online]. Available: http://www.blender.org

[13] G. Jocher, A. Chaurasia, A. Stoken, J. Borovec, NanoCode012, Y. Kwon, TaoXie, J. Fang, imyhxy, K. Michael, Lorna, A. V, D. Montes, J. Nadar, Laughing, tkianai, yxNONG, P. Skalski, Z. Wang, A. Hogan, C. Fati, L. Mammana, AlexWang1900, D. Patel, D. Yiwei, F. You, J. Hajek, L. Diaconu, and M. T. Minh, "ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference," Feb. 2022. [Online]. Available: https://doi.org/10.5281/zenodo.6222936

[14] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[15] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035. [Online]. Available: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf

[16] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, p. 674–679.

[17] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, 2008.

[18] S. Rusinkiewicz and M. Levoy, "Efficient variants of the icp algorithm," in *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE, 2001, pp. 145–152.

[19] K.-L. Low, "Linear least-squares optimization for point-to-plane icp surface registration," *Chapel Hill, University of North Carolina*, vol. 4, no. 10, pp. 1–3, 2004.

[20] S. Peng, Y. Liu, Q. Huang, X. Zhou, and H. Bao, "Pvnet: Pixel-wise voting network for 6dof pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[21] F. Engelmann, J. Stueckler, and B. Leibe, "SAMP: shape and motion priors for 4D vehicle reconstruction," in *IEEE Winter Conference on Applications of Computer Vision, WACV*, 2017.