2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

SAVOR: Sonar-Aided Visual Odometry and Reconstruction for Autonomous Underwater Vehicles*

Jeremy Paul Coffelt[1], Peter Kampmann[1], and Bilal Wehbe[2]

***Abstract*— Visual odometry (VO) relies on sequential camera images to estimate robot motion. For underwater robots, this is often complicated by turbidity, light attenuation, and environments containing scarce or repetitive features. Even ideal imagery suffers from the issue of scale ambiguity common to all monocular VO implementations. To address these issues, we supplement a camera with a multibeam echosounder. This acoustic, time-of-flight sensor comes with its own challenges, including relatively slow and sparse measurements that can be further degraded by backscatter from suspended particulate matter as well as interfering sounds from nearby marine traffic. We propose a method for fusing only data from these two inspection sensors into a hybrid VO solution that does not rely on IMU, DVL, or any other positioning sensor. We demonstrate this method on real data collected by an autonomous underwater vehicle performing end-to-end pipeline inspection in the open ocean, where multiple passes through the same scene (i.e., the "loop closure" common to SLAM algorithms) is often time and cost prohibitive. We also show how this approach can be extended for the creation of dense point clouds that provide a colored reconstruction of the surveyed scene.**

I. INTRODUCTION

With the growing interest in the blue economy and the utilization of the oceans as a source of sustainable resources, tasks conducted underwater, such as inspections of pipeline and cable networks, bathymetric surveys, and maintenance of offshore and subsea infrastructures are becoming increasingly vital from both environmental and economic standpoints. Although autonomous underwater vehicles (AUVs) have been a key technology for performing such tasks, their autonomy is significantly affected by localization challenges related to degraded perception, low-bandwidth communication, and denial of satellite-based positioning services (GNSS/GPS). These complications become even more apparent with the recent emergence of long-range, port-to-port AUV missions covering hundreds of kilometers per run [1].

When absolute position data is unavailable (e.g., for indoor aerial robots, ground robots in mine tunnels, and underwater robots), relative position can be estimated with a camera using either visual odometry (VO) or simultaneous localization and mapping (vSLAM). In SLAM, a robot attempts to build a map of its surroundings while simultaneously estimating its location in that map [2]. A key component of SLAM is loop closure, which involves first identifying points that have been previously visited and then using them to reduce accumulated errors in the global map. However, for many robot missions, including end-to-end pipeline/cable inspections, any such looping or retracing of trajectories is undesirable. Because of

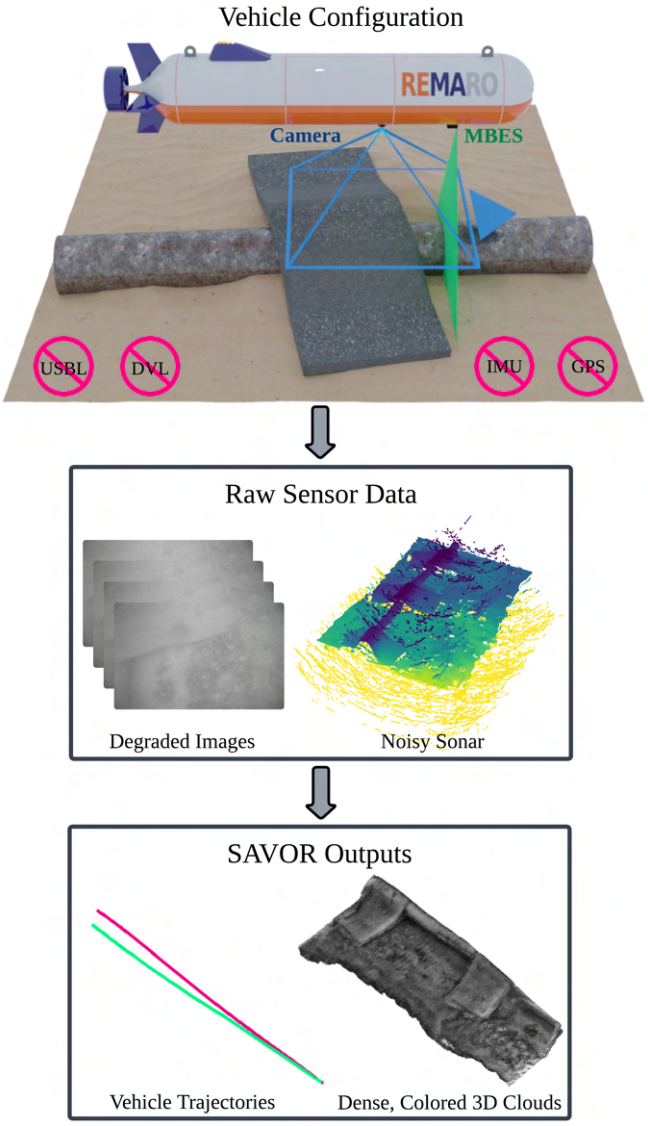

Fig. 1. SAVOR uses monocular imagery and MBES bathymetry to estimate vehicle odometry as well as dense, colored 3D reconstructions without relying on positioning sensors, such as an IMU, DVL, GPS, or USBL.

*This project has received funding from the European Union's Horizon 2020 research and innovation program under the Marie Skłodowska-Curie grant agreement No. 956200. Part of the work has also been funded by the CIAM project by the Ministry of Economics and Climate Action according to a decision of the German Bundestag.

[1]Jeremy Coffelt (corresponding author) and Peter Kampmann are with the Subsea Solutions group at Rosenxt in Bremen, Germany {jcoffelt, pkampmann}@rosen-nxt.com

[2]Bilal Wehbe is with the Robotics Innovation Center at the German Research Center for Artificial Intelligence (DFKI) in Bremen, Germany bilal.wehbe@dfki.de

this, VO is often the preferred approach for online tracking of relative camera/vehicle motion [3]. Rather than using rotary encoders to determine odometry, VO relies on the movement of distinctive features across consecutive images. By triangulating these tracked points, the relative motion of the camera can be iteratively estimated. Unfortunately, pure monocular VO suffers from a depth/scale ambiguity, which is often dealt with by using a second sensor, such as another camera (forming a stereo pair) [4] or a range sensor, such as LIDAR [5].

Underwater robots face additional challenges with both images and range sensors. For instance, turbidity, discoloration, and light absorption all contribute to poor visibility and degraded imagery [6]. Acoustic range sensors, such as multibeam echosounders (MBES), manage to overcome these difficulties, but instead suffer from lower spatial resolution and signal-to-noise ratios, which can be exacerbated by backscatter from suspended particles and interference from competing noise sources, such as nearby marine traffic and even other onboard sonar devices.

In this work, we make the following contributions to address many of the above challenges:

- A novel sonar-aided visual odometry (SAVO) algorithm ideally suited for vehicles (or towing rigs) with minimal sensor suites, as well as for salvaging survey data collected without (or with corrupted) positioning data.
- Efficient and reliable procedures for overcoming weaknesses of both sensors for improved feature tracking and depth estimation in underwater VO.
- An extension from SAVO to SAVOR for the generation of not only trajectories, but also dense, colored 3D reconstructions of surveyed scenes.
- Experiments with real AUV mission data on the effectiveness of the proposed VO and reconstruction routines for accurate, real-time performance without loop closure or other complications of full SLAM solutions.

II. RELATED WORK

VO and vSLAM are active and mature areas of research in ground and aerial robots, with notable contributions including [7] and [8], and excellent overviews in [9] and [10]. Here, we focus on underwater VO for the following reasons. First, the underwater domain tends to lag behind others in development. Second, our interest is primarily in linear missions, such as pipeline inspections, which seek to avoid revisiting the same scene multiple times and, therefore, benefit less from full SLAM implementations. Finally, we are interested in real-time solutions requiring both minimal processing power and sensor configurations, ideally using only two inspection sensors: an optical sensor to capture textures and a ranging sensor for geometry.

One of the earliest attempts at underwater monocular VO is given in [11], where texture analysis is employed to detect and track matches in consecutive image frames. A critical step in their approach and many others is the decomposition of the essential matrix to estimate relative translation and rotation between image frames [12]. As observed in [11] and our own experiments, the seafloor is often planar enough to result in degeneracy of the epipolar geometry assumptions underlying the essential matrix. In [13], seafloor planarity is instead exploited to compute and decompose the homography matrix for relative pose updates. In our own pipeline inspection field trials, we found the pipeline, mattresses, and seafloor resulted in scenarios where planar (homography) *and* non-planar (essential/fundamental matrix decomposition) approaches both proved unreliable. Some more recent

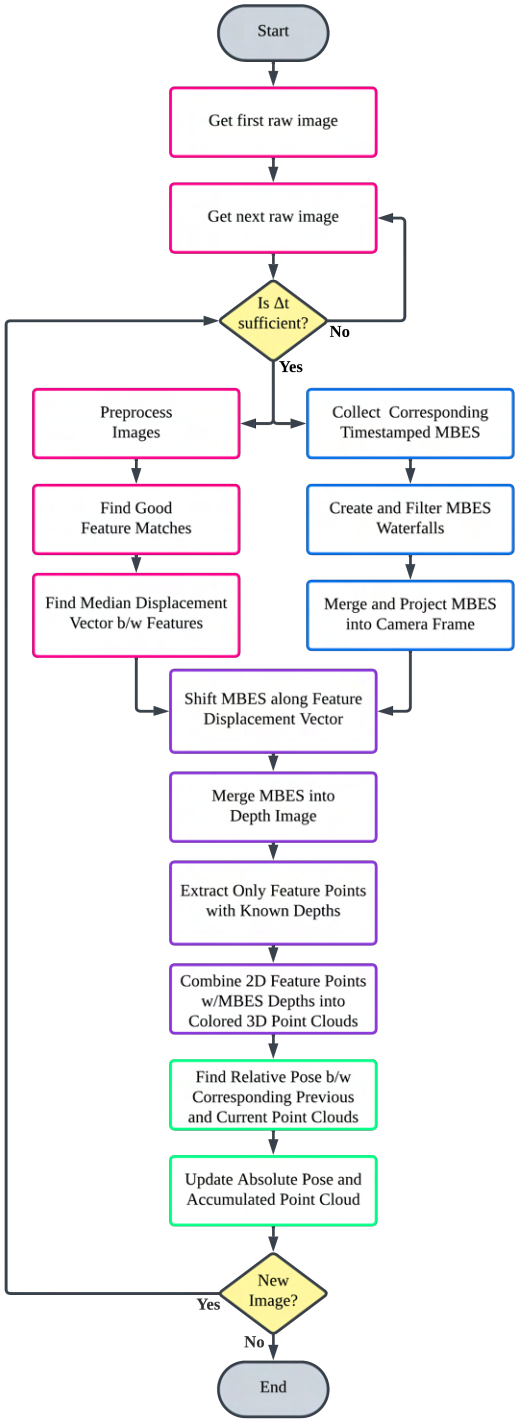

Fig. 2. Overview of the proposed SAVOR method. Processes involving only images are colored in magenta, only MBES in blue, both in violet, and trajectory/cloud outputs in green.

approaches, such as those tested in [14], use deep learning frameworks to bypass the above issues. In our work, we take a simpler and more direct approach, as detailed in Sec. III-C.

The methods described above suffer from the issue of scale ambiguity common to any monocular VO implementation [15]. The standard approach to resolving this issue is the usage of a second sensor, such as another camera (stereovision) or a time-of-flight range sensor. The former approach was used in [16] with a pair of RGB cameras. However, the works most similar to ours, including [17], [18], and [19], all used some sonar variant. In [17], a forward-facing single-beam echosounder is paired with an RGB camera on a remotely-operated vehicle (ROV) that was tested on multiple loops of an elliptical path roughly 20 m in circumference. In [18], a forward-looking imaging sonar is used for absolute scale estimation in a monocular VO framework demonstrated in simulation on trajectories about 25 m in length. Finally, in [19], scale estimation is provided by a forward-looking profiling sonar during a ship hull inspection approximately 15 m in length. In all of these works, vehicles were configured with forward-facing sensors. In that configuration, parallax/depth values can be reliably determined by features that streak radially outward in the image as the robot moves through its scene. Our experiments use a more difficult configuration, with a camera facing downward towards a largely planar seafloor, which produces image features following near-parallel tracks across consecutive images as the vehicle moves above the observed scene. Finally, our work investigates a real world trajectory that is an order of magnitude longer than those considered above.

III. METHODS

The end-to-end SAVOR procedure is summarized in Fig. 2 and described in detail below.

A. Image Preprocessing

Fig. 3 shows a sequence of preprocessing steps for raw images to improve feature detection. First, adaptive histogram equalization (AHE) is used to make images appear more uniformly lit both within each region of a given image, as well as across all images of an entire mission. More specifically, contrast-limited AHE (CLAHE) is chosen due to its ability to improve contrast while avoiding the problem of noise amplification common to many other AHE methods [20]. Next, the images are undistorted to correct for both radial and tangential distortion. Afterwards, images are reduced in size by 50% to decrease computational load. Finally, a median filter is applied to reduce the salt and pepper noise common to both underwater and low-light images. This preprocessing sequence is implemented in Python using the Open Computer Vision (OpenCV)[1] library.

B. MBES Preprocessing

In rough or turbid waters or near busy harbors, acoustic data such as multibeam bathymetry can be particularly noisy. To partially offset this, multibeam data can be separated

[1]https://opencv.org/

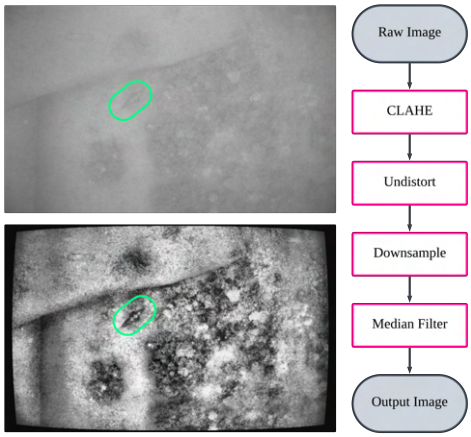

Fig. 3. Left: A typical image before (top) and after (bottom) preprocessing. The circled fish shows the improvements in both clarity and contrast. Right: The preprocessing sequence for each image.

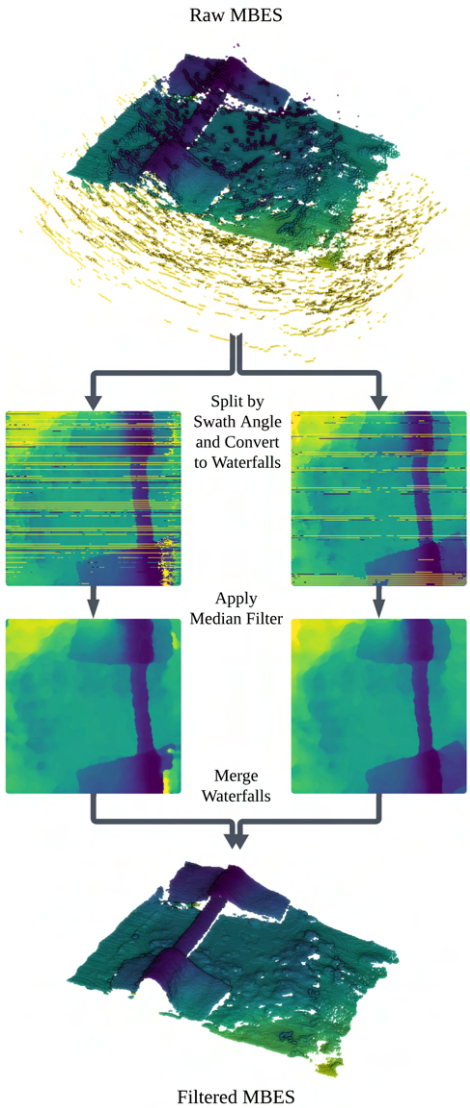

Fig. 4. MBES preprocessing procedure consists of splitting raw MBES by acoustic frequency / swath angle, applying a median filter to each waterfall image, and then merging back the filtered bathymetry.

by swath angle (if there is more than one). Then, for each angle, consecutive pings can be stacked into so-called "waterfall" images where the pixel value in row p column b corresponds to the bathymetry/range measurement of ping p and beam bearing b. These separate images can then be denoised using a median filter, and then merged back into a 3D cloud. An example of this procedure is shown in Fig. 4.

C. Image Features — Detection, Description, and Matching

For triangulation purposes, identifiable features must be detected, described, and matched in consecutive images. There are many popular approaches to this, including SIFT [21], SURF [22], ORB [23], and AKAZE [24]. A comparative analysis of these alternatives is given in [25], while an evaluation specifically targeted at their effectiveness in the underwater domain can be found in [26].

In this work, we focus primarily on ORB due to its speed, but we also consider SIFT and AKAZE. We observe that ORB tends to cluster on a particular texture (see Fig. 5 for a typical example), while the others seem to find features more uniformly distributed across the scene. Unfortunately, we also noticed that all methods struggle to find "good" matches in the repetitive textures and geometry common to the seafloor, rocks, and pipeline.

For ORB features, we first use a generous distance threshold in Lowe's ratio test [21] to eliminate the majority of false feature matches. However, as seen in the top right of Fig. 5, this approach alone is insufficient. Rather than using a stricter threshold, which occasionally eliminates virtually all candidate matches, we use an ad hoc approach for further filtering. The key observation is that the majority of vectors connecting a feature in the current frame to its counterpart in the previous frame have similar lengths and slopes. To minimize the effects of outliers, we first compute a median displacement vector between the current, c, and previous, p, frames:

$$\vec{d} = \langle \Delta u, \Delta v \rangle = \left\langle \text{med}(u_c^i - u_p^i),\ \text{med}\left(v_c^i - v_p^i\right) \right\rangle,$$

where $\left(u^i, v^i\right)$ are the pixel coordinates of feature i. We then eliminate all features whose displacement vector,

$$\vec{d^i} = \left\langle u_c^i - u_p^i, v_c^i - v_p^i \right\rangle,$$

has a length beyond some tolerance of the length of the median vector. Of those features that remain, we repeat a similar process by using the circular median [27] to remove features whose displacement vector has a slope sufficiently different than the slope of the median vector. These additional filtering steps proved reliable in keeping only the best feature correspondences. The right column of Fig. 5 shows the effectiveness of this approach in a typical image pair.

D. MBES Interpolation

Between each pair of consecutive images, timestamped MBES pings are accumulated. Each of these pings is first transformed from the MBES sensor coordinate frame through the AUV `base_link` to the camera sensor frame. These 3D coordinates are then projected into pixel coordinates using the camera projection matrix

$$P = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

where f and c are the focal lengths and principal point coordinates, respectively [12]. The full conversion from MBES $(x_{\text{mbes}}, y_{\text{mbes}}, z_{\text{mbes}})$ to pixel coordinates, (u, v), is then

$$P\left(T_{\text{base}}^{\text{cam}}\right)^{-1} T_{\text{base}}^{\text{mbes}} \begin{bmatrix} x_{\text{mbes}} \\ y_{\text{mbes}} \\ z_{\text{mbes}} \\ 1 \end{bmatrix} = z_{\text{cam}} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{1}$$

where T_A^B denotes the 4×4 homogeneous transformation matrix from coordinate frame A to B [28].

Performing this transformation for each of the 256 ranges in each ping results in a 2D trace across the pixel array. The pixels at these locations are then assigned a value proportional to the distance from the camera. In our case, we assume $0 = z_{\text{min}} \leq z_{\text{cam}} \leq z_{\text{max}} = 10$, which results in 8-bit grayscale values of the form

$$g = \text{round}\left(255 \cdot \frac{z_{\text{clip}} - z_{\text{min}}}{z_{\text{max}} - z_{\text{min}}}\right), \tag{2}$$

where

$$z_{\text{clip}} = \max\left\{z_{\text{min}}, \min\left\{z_{\text{max}}, z_{\text{cam}}\right\}\right\}.$$

Since the timestamp of the i^{th} ping, t^i, falls between the timestamps of the previous, t_p, and current, t_c, images, the i^{th} MBES trace is slid a proportional amount along the median feature displacement vector:

$$\left\langle u^i, v^i \right\rangle \rightarrow \left\langle u^i + k\Delta u, v^i + k\Delta v \right\rangle,$$

where

$$k = \frac{t_i - t_p}{t_c - t_p}.$$

Repeating this for all MBES pings collected between t_p and t_c results in a grayscale band in pixel coordinates corresponding to known depths, as shown in Fig. 6. (The viridis

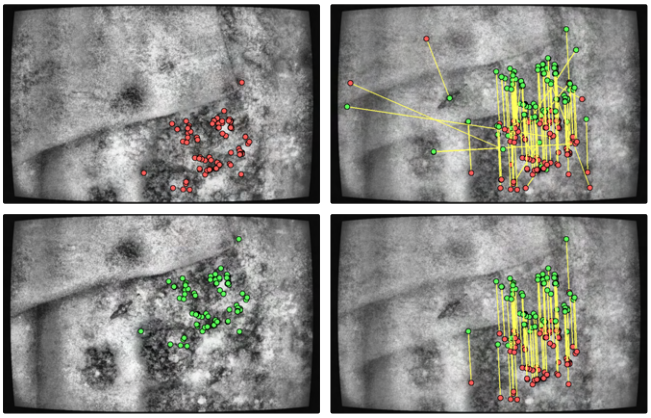

Fig. 5. Feature detection and matching between consecutive frames. Left: current (top) and previous (bottom) frames with ORB features. Right: Preliminary (top) and refined (bottom) "good" matches.

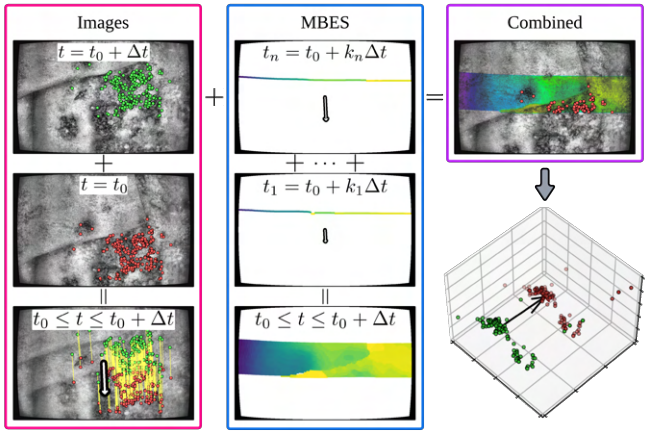

Fig. 6. Roles and interactions of camera and MBES data. Matched image features yield a displacement vector, which is used to properly place MBES pings in the image frame. Together, this information yields 3D points in consecutive frames, which can be used to estimate relative camera movement.

colormap is used in place of grayscale only so the depths can be seen when overlaid onto the original camera image.)

E. 3D Triangulation and Pose Updating

As shown in Fig. 6, only the good matches that lie within the projected and shifted MBES band are used to estimate the camera movement between frames. Suppose each of these matched 2D keypoints has pixel coordinates (u, v) and grayscale pixel value g. These pixel values are first converted into z-coordinates in the camera frame by inverting Eq. 2. With these z values, Eq. 1 can be used to produce corresponding 3D points. Repeating this for all feature points results in a sparse point cloud in each camera frame.

To find the transformation between these two point clouds, the centroid of each is first found. The translation vector, $\Delta\vec{t}$, is then estimated to be the displacement vector between these two centroids. Next, both point clouds are shifted so that their centroids share a common origin. Because of the assumed locations of the centers of mass and buoyancy, the vehicle considered here is prevented from all but negligible rolling and pitching. The final step is then to determine the relative yaw, $\Delta\theta$, between consecutive frames, which is done using an iterative approach to find the least squares error between the two shifted (denoted by the overlines) and rotated clouds

$$\underset{\Delta\theta}{\operatorname{argmin}} \sum_i \left(\begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) & 0 \\ \sin(\Delta\theta) & \cos(\Delta\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \bar{x}_c^i \\ \bar{y}_c^i \\ \bar{z}_c^i \end{bmatrix} - \begin{bmatrix} \bar{x}_p^i \\ \bar{y}_p^i \\ \bar{z}_p^i \end{bmatrix} \right)^2$$

With relative translation, $\Delta\vec{t}$, and yaw, $\Delta\theta$, estimated between frames, the new absolute location and rotation become

$$\vec{t}_{\text{new}} = \vec{t}_{\text{old}} + \begin{bmatrix} \cos(\theta_{\text{old}}) & -\sin(\theta_{\text{old}}) & 0 \\ \sin(\theta_{\text{old}}) & \cos(\theta_{\text{old}}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \Delta\vec{t} \quad (3)$$

and

$$\theta_{\text{new}} = \theta_{\text{old}} + \Delta\theta. \quad (4)$$

F. Dense Colored Point Cloud Reconstruction

So far, the only image points converted back into 3D are those of features with good matches that also fall within the projected MBES band. This results in a sparse point cloud with an initially unknown absolute location. A slight modification of this approach can also create a dense, colored point cloud with correct absolute coordinates. The first step is repeating the same process for all pixels with coordinates lying in the MBES band (not just those corresponding to matched features). This again involves inverting Eqs. 1-2. The grayscale intensity at each pixel in the MBES band provides a depth, which, in turn, yields a 3D point. The color of this 3D point is then assigned the same color as the pixel in the same location in the corresponding camera image. This results in a dense, colored point cloud with coordinates in the current camera frame. Applying the transforms in Eqs. 3-4 then places this incremental cloud in its correct absolute location. By merging all of these inter-frame clouds, a complete point cloud is produced for the entire mission. An example of such a cloud is shown in Fig. 11.

IV. EXPERIMENTS AND DISCUSSION

To demonstrate the proposed method, real data is evaluated from recent field trials of a pipeline inspection performed by a research-grade AUV in the Mediterranean Sea. This vehicle was equipped with standard positioning sensors, including a GNSS/GPS receiver, ultra-short baseline (USBL) transponder, depth/pressure sensor, Doppler velocity log (DVL), and inertial navigation system (INS) with built-in fiber-optic gyroscope (FOG) inertial measurement unit (IMU). During the mission, the AUV was fully submerged and unable to receive GNSS/GPS updates. In addition, rough waters and nearby marine traffic rendered the USBL ineffective. For these reasons, "ground truth" trajectories were produced by the INS using a fusion of measurements from the depth/pressure sensor, DVL, and IMU. In this configuration, the INS datasheet claims accuracies of $\pm 0.05\%$ of distance traveled and orientations within $0.025°$ deg RMS.

Images were produced by a camera configured to generate grayscale images of size 1936×1216 pixels at 20 Hz. The camera was positioned on the bottom of the AUV, oriented downward towards the nadir, between the MBES and center of mass/buoyancy. Due to the daytime mission and shallow waters in which the survey took place, images were sufficiently lit by ambient sunlight. A typical example of these raw images is shown at the top of Fig. 3.

Multibeam data was produced from pings at 45 Hz with alternating acoustic frequencies of 400 and 700 kHz and corresponding swath angles of $100°$ and $90°$, respectively. These swath angles are composed of 256 equally-spaced beams. The MBES was positioned on the bottom of the AUV, oriented downward towards the nadir, between the camera and nose of the vehicle.

With this hardware, we investigate the performance of the proposed method with varying image frequency, feature detectors/descriptors, and sonar configurations.

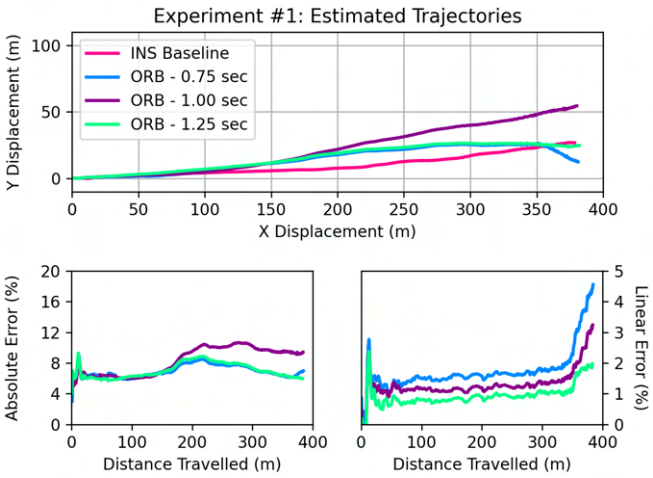

Fig. 7. Experiment #1: Effects of image frequency on estimated trajectories (top), absolute displacement error (left), and relative linear error (right).

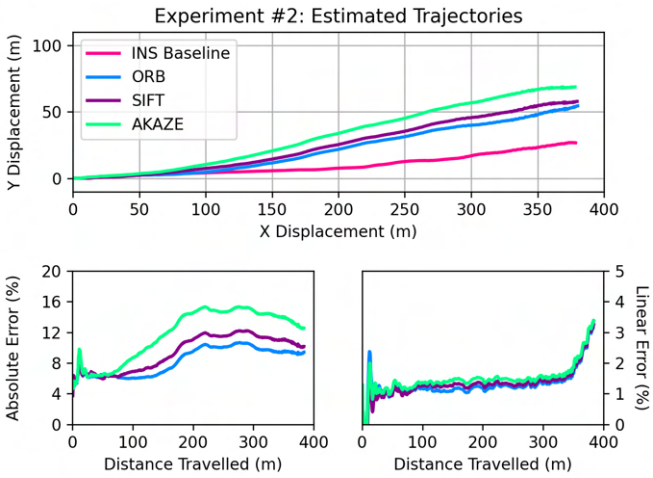

Fig. 8. Experiment #2: Effects of feature type on estimated trajectories (top), absolute displacement error (left), and relative linear error (right).

TABLE I

EXPERIMENT #1: IMAGE DELAY VS TRAJECTORY DEVIATIONS

Image Delay (s)	— Absolute Trajectory Errors — Mean (m)	RMSE (m)	Max (m)	Real-Time Factor
0.75	2.35	2.78	11.38	1.47
1.00	2.53	2.69	4.96	1.61
1.25	1.58	1.65	2.96	1.69

TABLE II

EXPERIMENT #2: FEATURE TYPE VS TRAJECTORY DEVIATIONS

Feature Method	— Absolute Trajectory Errors — Mean (m)	RMSE (m)	Max (m)	Real-Time Factor
AKAZE	2.90	3.12	7.31	1.54
ORB	2.53	2.69	4.96	1.61
SIFT	2.51	2.68	5.64	1.47

A. *Experiment #1: Image Frequency*

In VO, image frequency should be chosen to maximize the count and disparity of matched features. For the mission considered here, the AUV was configured to maintain a cruising speed of 0.7 m/s and a tracking altitude of 5 m. As a result, images processed around 1 Hz proved most effective even though images were available at 20 Hz. To investigate the sensitivity of the proposed sonar-aided visual odometry (SAVO) method on image frequency, ORB features were extracted and triangulated at intervals of 0.75, 1.00, and 1.25 s. As shown in the top of Fig. 7, all trajectories were quite close to the INS baseline. The bottom plots in Fig. 7 compare trajectory deviations against this baseline log distance (i.e., the distance traveled along the pipeline and forward axis of the vehicle). On the left is the absolute error between the baseline and SAVO-estimated coordinates, which shows a large initial error likely caused by the first few images being especially degraded as the vehicle made its initial descent towards the seafloor. Perhaps most interesting is the bottom right plot in Fig. 7, which shows the difference between baseline and SAVO remains small throughout the mission with negligible change, except at the ends of the mission when the AUV is diving or surfacing.

In Table I, absolute trajectory errors (ATEs) are computed using the approach described in [29] and implemented in the `evo`[2] Python package. As shown, at all tested intervals, mean ATEs are approximately 2 m or 0.5% of the total distance traveled. The last column in Table I provides a measure of how capable the particular configuration is of performing in real time. In particular, the minimum factor of 1.47 corresponds to a computation time 47% faster than real mission time:

$$\text{real-time factor} = \frac{\text{mission time}}{\text{processing time}} = \frac{534.9\ \text{s}}{364.3\ \text{s}} = 1.47$$

[2]https://github.com/MichaelGrupp/evo

B. *Experiment #2: Feature Detectors*

Next, we investigate effects of feature detector/descriptor choice on SAVO performance. For consistency, images were analyzed every 1 s. As shown in Fig. 8 and Table II, all approaches perform similarly, but with ORB performing quickest and with accuracy similar to SIFT and noticeably better than AKAZE.

C. *Experiment #3: Sonar Beam Count and Ping Frequency*

In order to quantify the benefits of MBES data over single-beam echosounders (SBES) for position estimation, our next experiment investigates the impacts of sonar frequency and beam count. We consider four scenarios, the first of which we will refer to as “All MBES”. It is the same scenario used in the previous experiments, where an MBES is available and all pings occurring between image frames are used for pose updates. The second, “One MBES”, uses only one MBES ping between consecutive frames. Third is “One SBES”, which assumes only an SBES is available and only provides one ping between image/pose updates. Finally, “All SBES” uses all available pings of an SBES that fall between pose updates. We simulate these configurations by using only part of the real MBES data collected during

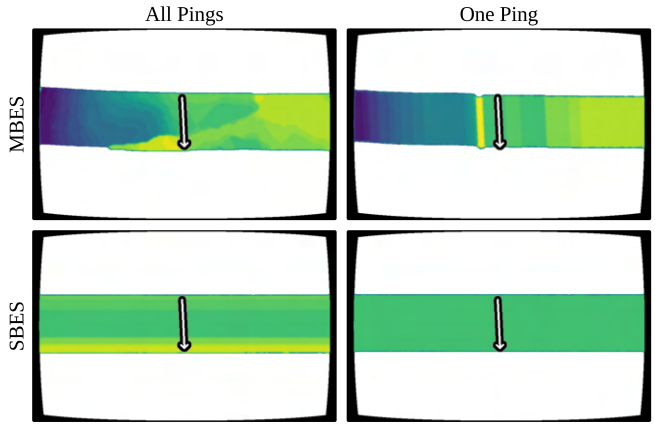

Fig. 9. Experiment #3: Depth bands (range measurements projected into the image frame) for various simulated sonar frequencies and beam counts.

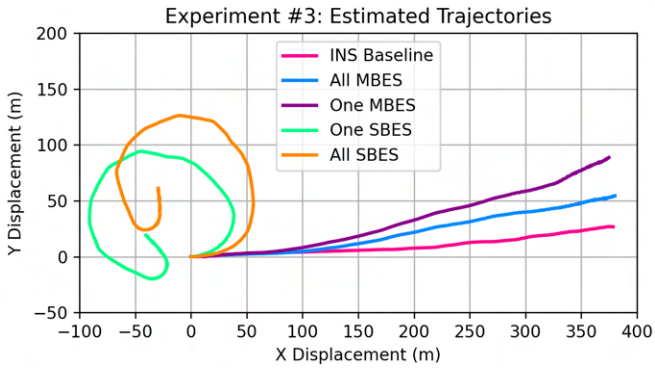

Fig. 10. Experiment #3: Effects of sonar beam count and ping frequency on estimated trajectories.

our field trials. That is, for the SBES variants, we use only a central beam of the 256 available in each MBES ping. And, for one-ping experiments, we simply use the ping with timestamp closest to the midpoint of the timestamps in consecutive image/pose updates. Example depth bands are shown in Fig. 9 with corresponding trajectories in Fig. 10.

As seen in Fig. 10, the proposed method can break down when used with an SBES. In our scenario, using all pings from an SBES amounts to approximating the cross-track depths, which primarily correspond to the seafloor, with a depth measurement that instead usually corresponds to the pipeline. Using only one SBES ping is even worse, as it amounts to approximating the current scene as a horizontal plane lying on top of the pipeline. For scenarios involving only seafloor surveys or at least inspections of cables and pipelines with much smaller diameters, the proposed method will likely perform much better. In any case, Fig. 10 also shows that the proposed method still performs reasonably well even when MBES data is only sampled at the much slower image-frame/pose-update frequency.

D. Experiment #4: Point Cloud Reconstruction

To demonstrate the reconstructive abilities of the SAVOR method, we apply the approach discussed in Sec. III-F to our field trial data. Recall this amounts to applying Eqs. 1-2 to *all* points in the combined RGBD image in Fig. 6 (not just the feature points with good matches lying within the MBES depth band). The result of this approach is shown in Fig. 11 alongside the corresponding raw MBES point cloud and a subsample of it processed as described in Fig. 4. These MBES clouds are based on the same INS baseline trajectory described in Experiment #1. The clouds on the left of Fig. 11 are 55 m long and correspond to 76 s of mission time, while the subsamples are 12 m / 17 s long. From this example, several observations can be made.

The first observation is that the SAVOR cloud is grayscale. This is only because the camera used in the mission was also grayscale. Full RGB clouds can easily be produced when color images are available by separately applying Eq. 2 to each color channel. Second, the MBES cloud covers a noticeably wider area of seafloor than the SAVOR cloud. This is because the SAVOR approach depends on *both* MBES pings and camera images, and, therefore, can only produce colored points where data from both sensors is known (or can be interpolated). The third observation is that the MBES cloud (before or after smoothing) is much sparser than the SAVOR cloud. For instance, in the subsamples, the MBES cloud contains 206k points, which corresponds to approximately 800 pings each containing 256 ranges. The corresponding SAVOR cloud contains 1.86 million points, which is approximately equivalent to the total number of pixels in three of the reduced-sized images processed as in Fig. 3 (or a single full-size raw image). The final observation is barely noticeable at the scales shown in Fig. 11, but each of the clouds exhibits varying amounts of jittering in the alignment between successive pings/frames. For the MBES clouds, this is less noticeable since updates are small (256 points per ping with pings occurring at 45 Hz) and frequent (pose updates from the INS baseline occur at 100 Hz). In contrast and by design, SAVOR updates are significantly larger and less frequent (with updates of roughly 150k pixels/points per second). This results in the SAVOR reconstruction being more jagged when zoomed in very close. However, this can be remedied some by reducing the pose/image update frequency, which may come at the cost of a slightly less accurate overall trajectory.

V. CONCLUSION

In this work, we propose SAVO, a sonar-aided visual odometry method that combines monocular imagery and MBES data to overcome common challenges of underwater localization, such as degraded imagery and scale ambiguity. We also introduce SAVOR, which is a straightforward extension for producing dense, colored 3D reconstructions.

To demonstrate the effectiveness of SAVOR, we conduct several experiments on real field trial data collected during a pipeline inspection performed by an AUV in the open ocean. Although the odometry is quite accurate, especially considering the perceptual challenges, we show that it is especially promising for scenarios involving pipeline and cable inspections, when absolute location is less important than the relative location (log distance) along the tracked pipeline/cable.

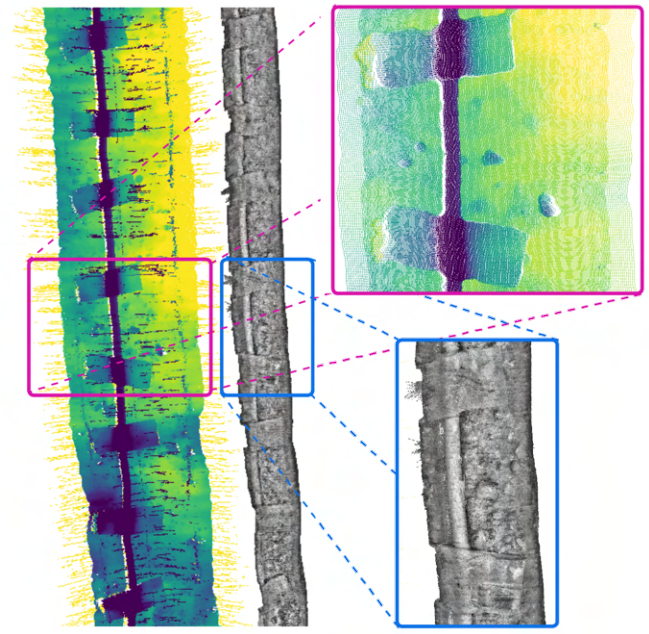

Fig. 11. Experiment #4: Comparison between sparse MBES cloud colored by range from sensor and dense SAVOR cloud colored by camera pixel values. The large MBES cloud contains raw sensor data, while the magenta closeup is filtered as described in Fig. 4. The SAVOR cloud is grayscale only because the camera was grayscale.

In future work, we plan to investigate several extensions to the current implementation. First, we plan to consider flow fields in place of single median displacement vectors for projecting MBES ranges collected during consecutive image frames. Next, we will examine increasing the delays between pose updates, with the intention of accumulating enough MBES measurements to produce a complete RGBD image, instead of the current version, which includes only a partial, inter-frame MBES depth band. Then, we hope to perform experiments with vehicles possessing a full six degrees of freedom versus the AUV considered in this work, which was restricted by design to four. We also hope to investigate the potential benefits towards global drift reduction from the addition of loop detection and closure. Lastly, we hope to adapt SAVOR to include inertial measurements from an IMU/DVL, when available, to overcome periods where images are either unavailable or are unusable.

References

[1] J. Zhou, Y. Si, and Y. Chen, "A review of subsea AUV technology," *Journal of Marine Science and Engineering*, vol. 11, no. 6, p. 1119, 2023.

[2] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[3] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proc. Conf. Computer Vision and Pattern Recognition (CVPR)*, vol. 1. IEEE, 2004, pp. I–I.

[4] A. Howard, "Real-time stereo visual odometry for autonomous ground vehicles," in *Int. Conf. Intelligent Robots and Systems (IROS)*. IEEE, 2008, pp. 3946–3952.

[5] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *Int. Conf. Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2174–2181.

[6] M. Ferrera, J. Moras, P. Trouvé-Peloux, and V. Creuze, "Real-time monocular visual odometry for turbid and dynamic underwater environments," *Sensors*, vol. 19, no. 3, p. 687, 2019.

[7] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Trans. on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[8] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2017.

[9] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *Robotics & Automation Magazine*, vol. 18, no. 4, pp. 80–92, 2011.

[10] A. Macario Barros, M. Michel, Y. Moline, G. Corre, and F. Carrel, "A comprehensive survey of visual SLAM algorithms," *Robotics*, vol. 11, no. 1, p. 24, 2022.

[11] R. Garcia, X. Cufi, and M. Carreras, "Estimating the motion of an underwater robot from a monocular image sequence," in *Proc. Int. Conf. Intelligent Robots and Systems (IROS)*, vol. 3. IEEE, 2001, pp. 1682–1687.

[12] R. Szeliski, *Computer Vision: Algorithms and Applications*. Springer Nature, 2022.

[13] S. S. da Costa Botelho, P. Drews, G. L. Oliveira, and M. da Silva Figueiredo, "Visual odometry and mapping for underwater autonomous vehicles," in *6th Latin American Robotics Symposium (LARS)*. IEEE, 2009, pp. 1–6.

[14] B. Teixeira, H. Silva, A. Matos, and E. Silva, "Deep learning for underwater visual odometry estimation," *IEEE Access*, vol. 8, pp. 44 687–44 701, 2020.

[15] R. Hartley and A. Zisserman, *Multiple-View Geometry in Computer Vision*. Cambridge University Press, 2003.

[16] J. Zhang, V. Ila, and L. Kneip, "Robust visual odometry in underwater environment," in *OES OCEANS Conf.* IEEE, 2018, pp. 1–9.

[17] M. Roznere and A. Q. Li, "Underwater monocular image depth estimation using single-beam echosounder," in *Int. Conf. Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 1785–1790.

[18] D. Yang, H. Ai, J. Liu, and B. He, "Absolute scale estimation for underwater monocular visual odometry based on 2D imaging sonar," *Measurement*, vol. 190, p. 110665, 2022.

[19] A. Cardaillac and M. Ludvigsen, "Camera-sonar combination for improved underwater localization and mapping," *IEEE Access*, 2023.

[20] S. Pizer *et al.*, "Adaptive histogram equalization and its variations," *Computer Vision, Graphics, and Image Processing*, vol. 39, no. 3, pp. 355–368, 1987.

[21] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int. Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[22] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Proc. European Conf. Computer Vision (ECCV)*. Springer, 2006, pp. 404–417.

[23] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Int. Conf. Computer Vision (ICCV)*. IEEE, 2011, pp. 2564–2571.

[24] P. F. Alcantarilla and T. Solutions, "Fast explicit diffusion for accelerated features in nonlinear scale spaces," *IEEE Trans. Pattern Analysis and Machine Intelligence (PAMI)*, vol. 34, no. 7, pp. 1281–1298, 2011.

[25] S. A. K. Tareen and Z. Saleem, "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK," in *Int. Conf. Computing, Mathematics and Engineering Technologies (iCoMET)*. IEEE, 2018, pp. 1–10.

[26] F. Hidalgo and T. Bräunl, "Evaluation of several feature detectors/extractors on underwater images towards vSLAM," *Sensors*, vol. 20, no. 15, p. 4343, 2020.

[27] K. V. Mardia and P. E. Jupp, *Directional Statistics*. John Wiley & Sons, 2009.

[28] P. I. Corke, W. Jachimczyk, and R. Pillat, *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer, 2011, vol. 73.

[29] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry," in *Int. Conf. Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 7244–7251.