DCNet: A Data-Driven Framework for DVL Calibration

Zeev Yampolsky* and Itzik Klein
The Hatter Department of Marine Technologies
Charney School of Marine Sciences, University of Haifa, Haifa, Israel

Abstract—**Autonomous underwater vehicles (AUVs) are underwater robotic platforms used in a variety of applications. An AUV's navigation solution relies heavily on the fusion of inertial sensors and Doppler velocity logs (DVL), where the latter delivers accurate velocity updates. To ensure accurate navigation, a DVL calibration is undertaken before the mission begins to estimate its error terms. During calibration, the AUV follows a complex trajectory and employs nonlinear estimation filters to estimate error terms. In this paper, we introduce DCNet, a data-driven framework that utilizes a two-dimensional convolution kernel in an innovative way. Using DCNet and our proposed DVL error model, we offer a rapid calibration procedure. This can be applied to a trajectory with a nearly constant velocity. To train and test our proposed approach a dataset of 276 minutes long with real DVL recorded measurements was used. We demonstrated an average improvement of 70% in accuracy and 80% improvement in calibration time, compared to the baseline approach, with a low-performance DVL. As a result of those improvements, an AUV employing a low-cost DVL, can achieve higher accuracy, shorter calibration time, and apply a simple nearly constant velocity calibration trajectory. Our results also open up new applications for marine robotics utilizing low-cost, high-accurate DVLs.**

I. INTRODUCTION

Autonomous underwater vehicles (AUVs) play a significant role in a variety of marine and underwater applications. For example, oceanographic surveys [1, 2], underwater structure inspection such as gas pipelines [3–5], and detection of oil leaks [6, 7]. During the AUVs mission, the navigation solution is critical for the mission's success [8, 9]. The navigation solution, which is the position, velocity, and orientation of the AUV, is commonly provided by inertial sensors, Doppler velocity log (DVL), acoustic positioning, and other sensors such as pressure sensors [10, 11].

The DVL is an acoustic sensor that utilizes the Doppler frequency shift effect. It transmits acoustic beams to the seafloor, which are then reflected back. By using the frequency shift, the DVL is able to calculate each beam's velocity and then to estimate the AUV velocity [12, 13].

The DVL measurements are subject to different types of error sources and noise that influence the measurement quality and accuracy [14, 15]. To cope with such effects, DVL calibration is performed prior to mission start. In such a process, the error terms are estimated and calibrated for [16, 17]. Commonly, the calibration process requires the AUV to sail at sea level while using a reference velocity sensor to determine the AUV's

ground truth (GT) velocity. Commonly, a global navigation satellite system (GNSS) receiver with real-time kinematics (GNSS-RTK) is used as a reference sensor since it provides centimeter-level accuracy [18].

During the calibration procedure, the AUV is required to follow long and complex trajectories, which in turn employ a nonlinear estimation filter, resulting in an overall complex calibration process [19]. Typically, the DVL calibration algorithms are model-based approaches employing estimation filters such as the Kalman filter, adaptive Kalman filter [20, 21], and the invariant extended Kalman filter [16]. Other approaches propose employing genetic algorithms for the DVL calibration [22].

Data-driven methods have been employed in AUV-related tasks with promising outcomes. For example, deep-learning frameworks have been used to estimated missing DVL beams in partial and complete outage scenarios [15, 23, 24]. Additionally, when all beams are available, the BeamsNet approach offers a more accurate and robust velocity solution using a dedicated deep-learning framework [25].

Recently, we proposed a data-driven method for DVL calibration [26]. There, a procedure that employs a simple constant velocity trajectory and a data-driven framework has demonstrated, on simulative data, effective calibration performance. In this research we leverage our initial work and propose a simple, yet effective, DVL calibration process. To meet this goal we make the following contributions:

1) **DVL error models**: Five error models are proposed and investigated to relate the measured DVL velocity to the reference GNSS-RTK velocity. Each proposed error model utilizes a different combination of the DVL error terms including scale factor and bias.

2) **DCNet**: A simple, yet efficient, end-to-end regression network for the DVL calibration process is developed. DCNet is a convolution based framework that employs a two-dimensional dilated convolution kernel designed to process the input data in a unique manner.

3) **Data-driven calibration**: We propose a data-driven framework for the calibration process. Utilizing, our proposed error models and DCNet, our framework can be applied on a simple nearly constant velocity trajectory and requires less time for the calibration process.

To demonstrate our proposed approach, we employed a dataset consisting of 276 minutes. The data was recorded using the

*Corresponding author: Zeev Yampolsky, zyampols@campus.haifa.ac.il.

University of Haifa's AUV, the "Snapir" [25, 27], in two separate experiments totaling in ten recorded different trajectories. Our proposed data-driven approach achieved an average of 70% accuracy improvement, and a 80% improvement in calibration time, over the baseline approach on a low-grade DVL, using real-world recorded data. As a result of our work, low-cost DVLs can be used to achieve higher accuracy and reduce costs. This will open up new applications for marine robotic systems using low-cost DVLs with high accuracy.

The rest of this paper is organized as follows: Section II describes the DVL velocity measurement and the baseline calibration approach. Section III presents our proposed data-driven approach. Section IV gives the experimental results and Section V gives the conclusions of this research.

II. PROBLEM FORMULATION

A. DVL Velocity

For a DVL with an "$\times$" configuration, referred to as the "Janus" configuration, the direction of each beam in the DVL's body frame can be expressed as [25, 28]:

$$\boldsymbol{b}_i = \begin{bmatrix} \cos\psi_i \sin\alpha & \sin\psi_i \sin\alpha & \cos\alpha \end{bmatrix}_{1\times3} \qquad (1)$$

where ψ_i and α refer to the pitch and yaw angles of beam $i = 1, 2, 3, 4$, respectively. For all beams, the pitch angle remains constant, whereas the yaw angle is given by

$$\psi_i = (i-1) \cdot \frac{\pi}{2} + \frac{\pi}{4} \; [rad] \, , \; i = 1, 2, 3, 4. \qquad (2)$$

By stacking all beam projection vectors from (1), the transformation matrix $\mathbf{H} \in \mathbb{R}^{4\times3}$ is formed:

$$\mathbf{H} = \begin{bmatrix} \boldsymbol{b}_1 \\ \boldsymbol{b}_2 \\ \boldsymbol{b}_3 \\ \boldsymbol{b}_4 \end{bmatrix}_{4\times3} \qquad (3)$$

The estimated velocity vector, expressed in the DVL frame, is

$$\boldsymbol{v}_{\mathrm{AUV}}^d = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T\boldsymbol{y} \qquad (4)$$

The DVL error model is given by [12]:

$$\tilde{\boldsymbol{y}} = [\mathbf{H}\boldsymbol{v}_{\mathrm{AUV}}^d(1 + s_{\mathrm{DVL}})] + \boldsymbol{b}_{\mathrm{DVL}} + \boldsymbol{\sigma}_{\mathrm{DVL}} \qquad (5)$$

where $\tilde{\boldsymbol{y}}$ is the measured beam velocity vector, $\boldsymbol{v}_{\mathrm{AUV}}^d$ is the AUV velocity expressed in the DVL frame, s_{DVL} is the scale factor applied over the beams, $\boldsymbol{b}_{\mathrm{DVL}}$ is the additive bias vector, and $\boldsymbol{\sigma}_{\mathrm{DVL}}$ is the additive zero mean Gaussian white noise vector.

B. GNSS Based DVL Calibration

During the calibration process the AUV follows a predetermined trajectory while sailing at sea level, allowing GNSS-RTK velocity measurements of the AUV, while in parallel maintaining sufficient depth for the DVL to provide measurements. The GNSS-RTK and DVL measurements are related using the following error model [16]:

$$\hat{\boldsymbol{v}}^d = (1+k)\mathbf{R}_b^d(\mathbf{R}_n^b\boldsymbol{v}^n + \boldsymbol{\omega}_{nb}^b \times \boldsymbol{l}_{\mathrm{DVL}}) + \boldsymbol{\delta v}^d \qquad (6)$$

where $\boldsymbol{v}^n$ is the AUV reference velocity provided by the GNSS-RTK in the navigation frame, $\mathbf{R}_n^b$ is the transformation

matrix from the navigation to the body frame, $\boldsymbol{\omega}_{nb}^b$ is the angular velocity of the AUV, and $\boldsymbol{l}_{\mathrm{DVL}}$ is the lever-arm between the DVL and the AUV center of mass. Matrix $\mathbf{R}_b^d$ is the constant and known transformation matrix from the DVL to the body frame, k is the scale factor applied over the reference velocity in the DVL frame, and $\boldsymbol{\delta v}^d$ is the additive zero mean Gaussian white noise. Note that the term $\boldsymbol{\omega}_{nb}^b \times \boldsymbol{l}_{\mathrm{DVL}}$ is usually neglected since the lever-arm is generally small and can be measured and compensated [21]. Thus, (6) reduces to:

$$\hat{\boldsymbol{v}}^d = (1+k)\mathbf{R}_n^d\boldsymbol{v}^n + \boldsymbol{\delta v}^d \qquad (7)$$

where $\mathbf{R}_n^d$ is the transformation matrix from the navigation from to the DVL frame. By applying a vector norm on the error model presented in (7), the following is achieved [29]:

$$\| \hat{\boldsymbol{v}}^d \| = (1+k) \| \hat{\mathbf{R}}_n^d\boldsymbol{v}^n \| . \qquad (8)$$

Further development of (8) produces the following scalar scale factor estimation:

$$\hat{k}_t = \frac{\| \hat{\boldsymbol{v}}_t^d \|}{\| \hat{\mathbf{R}}_n^d\boldsymbol{v}_t^d \|} - 1, \; t = 1, \dots, T \qquad (9)$$

where $\hat{k}_t$ is the estimated scalar scale factor at time step t that is applied to the measured DVL velocity as presented in (7). Note that $\hat{k}_t$ is 1) a scalar scaling factor, thus the same scaling factor is applied over all three velocity axes, and 2) is an estimate at a single time step t, out of the entire calibration recording with T time steps. To estimate a single scaling factor and mitigate the zero mean Gaussian white noise effect, the average of all T time steps of $\hat{k}_t$ is calculated as follows:

$$\overline{k} = \frac{1}{T}\sum_{t=1}^{T}\hat{k}_t. \qquad (10)$$

This is the baseline approach, which provides the estimated scalar scale factor at the end of the calibration procedure.

III. PROPOSED APPROACH

In this section we present five different DVL error models and the derivation of our DCNet approach.

A. DVL Error Models

In our preliminary work [26] we proposed the following DVL error model:

$$\hat{\boldsymbol{v}}^d = (1 + \boldsymbol{k}_{\mathrm{DVL}})\hat{\mathbf{R}}_b^d\boldsymbol{v}^n + \boldsymbol{b}_{\mathrm{DVL}} + \boldsymbol{\delta v}^d \qquad (11)$$

where $\boldsymbol{k}_{\mathrm{DVL}}$ is the scale factor vector:

$$\boldsymbol{k}_{\mathrm{DVL}} = [k_x \; k_y \; k_z]^T \in \mathbb{R}^3 \qquad (12)$$

$\boldsymbol{b}_{\mathrm{DVL}}$ is the bias vector:

$$\boldsymbol{b}_{\mathrm{DVL}} = [b_x \; b_y \; b_z]^T \in \mathbb{R}^3 \qquad (13)$$

and $\boldsymbol{\delta v}^d$ is the zero mean Gaussian white noise. Note that, as presented in (12) and (13), the scale factor and bias error terms can be modeled as scalars or vectors. When only a single type of error term is presented (bias or scale factor), four different DVL error models can be deduced from (11), as we presented in [26].

With a goal to estimate the velocity error in each axis, we propose a comprehensive error model including a scale factor vector and a bias vector. Thus, in total, we consider five different DVL error models (EMs):

1) **Scale Factor Only:** This model corrects DVL measurements using only a scale factor, as commonly addressed in the literature. Two alternatives are considered for the scale factor-only model:

   a) **EM1 - Scalar Scale-Factor**: The scale factor has the same value $k_x = k_y = k_z = k$ in all three axes, and thus is addressed as a scalar.

   b) **EM2 - Vector Scale-Factor**: The scale factor values may differ in each axis as defined in (12).

2) **Bias Only:** The bias error model corrects DVL measurements using only a bias. Two alternatives are considered for the bias-only error model:

   a) **EM3 - Scalar Bias**: The bias has the same value $b_x = b_y = b_z = b$ in all axes, and thus is addressed as a scalar.

   b) **EM4 - Vector Bias**: The bias values may differ in each axis as defined in (13).

3) **EM5 - Vector Error Model**: Both scale factor and bias are vectors, allowing different combinations of scale factor and bias value for each axis, as presented in (12) and (13).

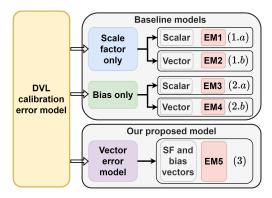Figure 1 presents the five EMs considered in this work.

Fig. 1: Five DVL error models divided into baseline models **EM1** - **EM4**, and our proposed **EM5**.

B. Proposed DVL Calibration Neural Network

Our proposed approach, DVL calibration neural network (DC-Net), is a data-driven approach utilizing a deep learning (DL) neural network (NN) framework to estimate the error terms in the different **EM1** - **EM5** as presented in Section III-A. Our NN, presented in Figure 2, is suitable for all EMs. This is a multi-head network with input, output, and three major parts, as described below:

1) **Input Size**: The input to the network is the same as the input to that presented in (7); that is the DVL and GNSS-RTK velocity vectors that were transformed to body frame, denoted as $\tilde{v}_{\text{DVL}}^b$ and $\tilde{v}_{\text{GNSS}}^b$, respectively. The input size is $< batch\ size \times 6 \times window\ size >$, where 6 is the number of stacked velocity axes, which are the DVL and GNSS-RTK XYZ axes, and 10 is the selected window size that is described in subsection IV-A.

2) **1DCNN Head**: The input velocity vectors are subtracted to produce v_{sub}^b, which has has only three axes, X, Y, and Z. The subtraction vector goes through two 1D convolution layers each followed by a LeakyReLU activation function (21). The vector v_{sub}^b is the subtraction of the GNSS-RTK from the DVL as follows:

$$\tilde{v}_{\text{sub}}^b = \tilde{v}_{\text{DVL}}^b - \tilde{v}_{\text{GNSS}}^b. \tag{14}$$

We used the subtraction, because, if we consider a scalar scale factor error model such as **EM1**, we can use (7) as follows:

$$\tilde{v}_{\text{sub}}^b = (1 + k_{\text{DVL}})\mathbf{R}_n^b v^n - \tilde{v}_{\text{GNSS}}^b. \tag{15}$$

At this point we ignore $\boldsymbol{\sigma} v^d$ since it can be averaged out. Recall that we use the GNSS-RTK as the reference and we assume $\hat{\mathbf{R}}_n^b = \mathbf{I}_3$, thus resulting in

$$\hat{\mathbf{R}}_n^b v^n = \tilde{v}_{\text{GNSS}}^b. \tag{16}$$

By using (16) with (15) we derive the following:

$$\begin{aligned}
\tilde{v}_{\text{sub}}^b &= (1 + k_{\text{DVL}})\tilde{v}_{\text{GNSS}}^b - \tilde{v}_{\text{GNSS}}^b \\
&= \tilde{v}_{\text{GNSS}}^b + k_{\text{DVL}} \cdot \tilde{v}_{\text{GNSS}}^b - \tilde{v}_{\text{GNSS}}^b. \\
&= k_{\text{DVL}} \cdot \tilde{v}_{\text{GNSS}}^b
\end{aligned} \tag{17}$$

Considering that we are using several training trajectories, we expect that the FC batch will be able to extract the error terms more easily when provided with the concatenated outputs of the two CNN heads, given the derived input (15) to the 1DCNN.

3) **2DCNN Head**: Is passed with the two velocity vectors, the GNSS-RTK and the DVLs, as presented in the input, through three 2D convolution layers, each followed by a LeakyReLU activation function (21). The first convolution's layer kernel is dilated by 3×1, while the other two are dilated by 1×1. This way, instead of processing the extracted features by the 1D convolution layers only once concatenated and passed through the FC batch, using the 2D convolution kernel allows us to extract features and process information stored in both vectors from the start through the first convolution layer. The dilated kernel in the first 2D convolution layer allows the first layer to process simultaneously the corresponding axes of the DVL and GNSS-RTK velocity; i.e., process both X axes. The motivation for the dilation is to enable the network to mimic the information extraction as in the subtraction vector, $\tilde{v}_{sub}^b$. Figure 3 illustrates the dilation versus the non-dilated kernel.

4) **FC Batch**: consists of four FC layers. Each layer of the first three is followed by a LeakyReLU activation function (21) and then by a dropout layer III-C4. The

final layer directly outputs the error terms according to the evaluated error model, **EM1** - **EM5**. The outputs of both convolution heads are flattened and then stacked, or concatenated over each other to create the input to the FC batch.

5) **Output**: The output itself and its size are determined by which error model **EM1-EM5** is being evaluated; i.e., if **EM2** is evaluated, the output would be a $\hat{s} \in \mathbb{R}^3$.

Next, we provide a mathematical formulation of our DCNet, describe the training process, and provide the implementation details. All the NN hyper parameters such as the batch size, the dropout probability, and activation function formulations are described in Section III-C and the values are presented in Table I.

C. DCNet Formulation and Training Process

In this section, we will examine the mathematical formulation of DCNet and the training process.

1) Fully Connected Neural Network: Neurons are the building blocks of a NN, where each neuron in the FC layer is defined by [25]:

$$z_i^{(l)} = \sum_{j=1}^{n_{(l-1)}} \omega_{ij}^{(l)} a_j^{(l-1)} + b_i^{(l)} \tag{18}$$

where $\omega_{ij}^{(l)}$ is the learnable weight of the ith neuron in the lth layer that is connected to the jth neuron in the $(l-1)^{th}$ layer. The term $b_i^{(l)}$ is the bias of the ith neuron in the lth layer, and $a_j^{(l-1)}$ is the output of the jth neuron in the $(l-1)^{th}$ layer. Each convolution layer is structured in a similar way where each kernel sized $m_1 \times m_2$ contains $m_1 \times m_2$ neurons.

2) Convolution Neural Network: CNNs are constructed similarly to a FC layer, and expand the single neuron (18) to a kernel (filter) of size $m_1 \times m_2$ neurons [25]:

$$C_{ij}^{(l)} = \sum_{\alpha=0}^{m_1} \sum_{\beta=0}^{m_2} \omega_{\alpha\beta}^{(r)} a_{(i+\alpha)(i+\beta)}^{(l-1)} + b^{(r)} \tag{19}$$

where $C_{ij}^{(l)}$ is the output of the kernel, and $\omega_{\alpha\beta}$ is the weight of the kernel in the (α, β) position in the rth convolution layer. The term $a_{(i+\alpha)(i+\beta)}^{(l-1)}$ is the output of the preceding convolution layer, and $b^{(r)}$ is the bias of the rth convolution layer.

3) Nonlinear Activation Functions: Traditionally, each neuron's output, whether of a FC layer (18) or a convolution layer (19), passes through a nonlinear activation function, otherwise a NN acts as a linear regression model [30]:

$$a_i^{(l)} = h(z_i^{(l)}) \tag{20}$$

where $z_i^{(l)}$ is the ith neuron in the lth layer output—in this case a FC layer (18)—h is the activation function, and a_i^l is the output of the activation function employed. In this work we use the following nonlinear activation functions:

1) **Leaky Rectified Linear Unit (LeakyReLU)**: An activation function that scales negative values and does not change positive values [30]:

$$LeakyReLU(C_{ij}^{(l)}x) = \begin{cases} C_{ij}^{(l)}, & \text{if } C_{ij}^{(l)} \geq 0 \\ \alpha \cdot C_{ij}^{(l)}, & \text{otherwise} \end{cases} \tag{21}$$

where α is the scaling factor of the negative value, and $C_{ij}^{(l)}$ is the input to the function, in this case a convolution layer kernel output (19) since LeakyReLU is applied over the convolution layers output both in the 1DCNN and the 2DCNN heads.

2) **Hyperbolic Tangent Function (Tanh)**: A continuous and differentiable nonlinear activation function, which outputs values within a defined range of $[-1, 1] \in \mathbb{R}$ as in [30]:

$$Tanh(z_i^{(l)}) = \frac{e^{z_i^{(l)}} - e^{-z_i^{(l)}}}{e^{z_i^{(l)}} + e^{-z_i^{(l)}}} \tag{22}$$

where e is the Euler constant, and $z_i^{(l)}$ is the FC layer output, which is the input to this activation function. As described in Section III, the TanH activation function is applied over the first three layers in the FC batch.

4) Regularization: During training, overfitting occurs when the NN learns noise patterns present in training data, which results in a low loss value during training but fails to generalize and performs poorly on the test data [31]. In this work, we use dropout regularization [32] to mitigate the overfitting problem. With dropout, some neurons are not used during training with probability $1 - p$, as follows [32]:

$$\begin{aligned} r_i^l &\sim Bermoulli(p) \\ \tilde{a}_i^l &= r_i^l \cdot a_i^l \\ a_j^{l+1} &= \omega_{ji}^{(l+1)} \tilde{a}_i^{(l)} + b_j^{(l+1)} \end{aligned} \tag{23}$$

where r_i^l is a random variable sampled from a $Bernoulli$ distribution with p probability, a_i^l is the activation function output over the ith neuron (20), and the term $\tilde{a}_i^l$ is the dot product of r_i^l and a_i^l. The dot product, $\tilde{a}_i^l$, is fed as input to the jth neuron in the $(l+1)^{th}$ layer after dropout was applied. The terms $\omega_{ji}^{(l+1)}$ and $b_j^{(l+1)}$ are the weights and biases as described in Section III-C1.

5) Training Process: The training process aims to determine the weights and biases of all the neurons comprising the NN, in order to achieve the lowest possible loss function score. The loss function used in this study is the mean squared error (MSE) loss function:

$$J(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) = \frac{1}{N} \| \boldsymbol{y}_i - \hat{\boldsymbol{y}}_i \|^2 \tag{24}$$

where $J()$ is the MSE function, $\boldsymbol{y}_i$ is the GT value, $\hat{\boldsymbol{y}}_i$ is the network estimation, and N is the number of samples. Forward passing refers to the process of passing the input through the network's various layers, such as (18) and (19). Based on the output of the network, the loss function value is calculated and derived with respect to the weights and biases of the NN. As part of the training process, the derivations are used to minimize the loss value in the next forward pass, by updating
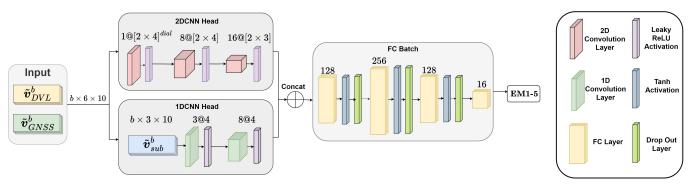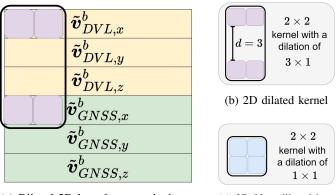
Fig. 2: Block diagram of our DCNet architecture used to estimate one of the proposed **EM1-EM5** error terms.

(a) Dilated 2D kernel process both velocity vectors

(b) 2D dilated kernel

(c) 2D Non-dilated kernel

Fig. 3: Illustration of two 2D convolution kernels sized 2×2 and how the dilated kernel processes both GNSS and DVL velocity vectors. Figure 3a shows the 3×1 dilated 2D convolution kernel processes both X axes of the DVL and GNSS velocities simultaneously. Figure 3b illustrates the 3×1 dilated 2D kernel, while Figure 3c shows a 1×1 2D kernel, which in fact is referred to as not dilated.

the weights and biases of the model based on those derivations. To accomplish this, a gradient descent algorithm is used:

$$\boldsymbol{\theta} = \boldsymbol{\theta} - \eta \nabla_\theta J(\theta), \quad \boldsymbol{\theta} = [\omega \ \ b]^T \qquad (25)$$

where $\boldsymbol{\theta}$ is the vector of the weights and biases of a neuron, η is the learning rate that is set at the start of the training process, $J(\boldsymbol{\theta})$ is the loss function value with respect to the vector of the weights and biases $\boldsymbol{\theta}$, and ∇_θ is the gradient operator with respect to the vector $\boldsymbol{\theta}$. To this end, a root mean squared propagation (RMSProp) optimizer is used [33]. We set the learning rate of the proposed NN to $5 \cdot 10^{-5}$ when training it to estimate the error terms of **EM1-EM4**, and to $5 \cdot 10^{-4}$ when training it to estimate the error terms of **EM5**. Note that the network estimates the error terms for one of **EM1-EM5** based on the error model that is being evaluated. The loss value, however, is calculated on the basis of velocities rather than error terms themselves. Following the network estimation of the error terms, the input DVL velocity is calibrated using the estimated error terms, and using both the input and calibrated values, the calculated loss value is used to update the weights.

Accordingly, the MSE loss using two velocities is as follows:

$$MSE(\boldsymbol{y}_i, \hat{\boldsymbol{y}}_i) = \frac{\sum_{i=1}^{N} [\sum_{j}^{\{X,Y,Z\}} (\boldsymbol{y}_{i,j} - \hat{\boldsymbol{y}}_{i,j})^2]}{N} \qquad (26)$$

where $\boldsymbol{y}_i \in \mathbb{R}^{N \times 3}$ is the GT velocity vector, $\hat{\boldsymbol{y}}_i \in \mathbb{R}^{N \times 3}$ is the calibrated velocity vector, i indicates the length of the velocity vector—that is, N—and j is the three axes X, Y, and Z. Figure 4 illustrates a block diagram of the forward pass and the input to the closed loop MSE loss function.
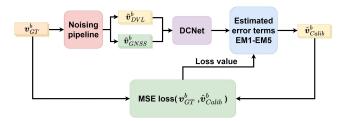

Fig. 4: Block diagram showing the closed loop MSE loss calculation procedure after the proposed approach estimates the error terms.

For the purpose of this section—that is, the weights and biases updated during training—the output of the MSE loss function is sufficient. To prevent the proposed NN approach from overfitting the data, a dropout regularization is used between the first three layers in the FC head as presented in Figure 2. All the hyperparameters for the training procedure described in Section III-B and fully formulated in this section are presented in Table I.

TABLE I: Hyperparameters used during the training process of our proposed NN approach.

Parameter	Epochs	B. size	2D Leaky	1D Leaky	Dropout p
Value	100	256	0.05	0.05	0.3

IV. EXPERIMENTAL RESULTS

A. Data Collection and Processing

To validate our proposed approach, data recorded using the University of Haifa's "Snapir" AUV was used. This data was provided in [25] and [34] and can be found at their associated GitHub repositories. Figure 5 shows an image of the AUV during a mission. "Snapir" is a torpedo-shaped AUV

measuring 5.5 meters in length and 0.5 meters in diameter. The "Snapir" employs a Teledyne RD Instruments Work Horse Navigator DVL, with a sampling rate of 1Hz and accuracy of 0.6 cm/s while traveling at 3 m/s [14].

Fig. 5: The "Snapir" AUV being pulled out of the water after a mission.

A total of ten trajectories were recorded in two separate sea experiments. In the first, nine trajectories, referred to as T1 - T9, were recorded with a total time of 243 minutes. In the second experiment, an additional trajectory was recorded (referred to as T10) with a total time of 33 minutes of recordings. Figure 6 presents the AUV velocity, expressed in the body frame, as a function of time during the T4 and T6. Next, trajectory T10 was divided into two parts. The first 200 seconds were allocated for evaluating the calibration parameters based on our trained network and are referred to as the calibration dataset T10Cal. The remaining 1800 seconds of T10 are part of the test dataset and are referred to as T10Test. The training dataset contains of trajectories T1-T5 while the test datasets includes trajectories T6-T9 and T10Test. The motivation for this selection is twofold: 1) T10 is the trajectory recorded in a different sea experiment, thus making it the calibration dataset aims to generalize and increase our model robustness as we train on trajectories from a different sea experiment. 2) T10Cal represents a simple nearly constant velocity trajectory, thus if we are able to learn the calibration parameters solely on T10Cal, we also offer a simple effective trajectory for the calibration process.

All the recordings contain the DVL measured velocity. To produce the sensor noise, we applied a moving average (MA) filter over all ten trajectories. We used a MA filter with a five second window [35]:

$$\boldsymbol{x}'_i = \frac{1}{T} \sum_{j=i}^{i+5} \boldsymbol{x}_j \tag{27}$$

where T is the total recording length, $\boldsymbol{x}_j$ is the original velocity vector at time step j, and $\boldsymbol{x}'_i$ is the MA filtered velocity vector. The filtered data is then referred to as our ground-truth (GT) data.

To produce the low-cost DVL measurements and the reference GNSS-RTK, we employed a noising pipeline presented in Figure 7, which takes as input the GT velocity. The noising pipeline is designed to mimic the beams error model (5),

(a) T4 Velocity Recording
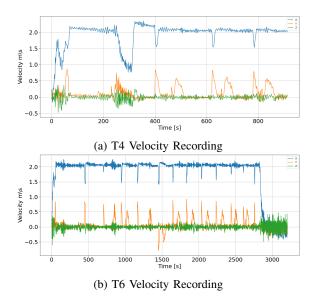

(b) T6 Velocity Recording

Fig. 6: AUV velocity as measured by the DVL during T4 and T6 trajectories.

transformation to the DVL frame (4), and later to the body frame. Additionally, the pipeline outputs the GNSS-RTK measurements by applying a zero mean Gaussian white noise to the GT measurements as follows:

$$\tilde{\boldsymbol{v}}^b_{\text{GNSS}} = \boldsymbol{v}^b_{\text{GT}} + \boldsymbol{\sigma}_{\text{GNSS-RTK}} \tag{28}$$

where $\boldsymbol{\sigma}_{\text{GNSS-RTK}} \sim N(0, 0.005\text{m/s}^2)$ and remains constant whenever the pipeline is used. Note that s_{DVL}, b_{DVL}, and σ_{DVL} are the only parameters that affect the noising pipeline outcome.
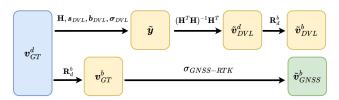

Fig. 7: The pipeline used to generate noisy GNSS and DVL velocity measurements based on the GT velocity from the sea experiments.

To create a sufficient sized train dataset that captures a wide range of DVL measurements characteristics, 1134 combinations of the error terms were used with the pipeline. Table II presents the error terms values and combinations that were added to the train trajectories T1 - T5. After adding the error terms, the trajectories were split into windows of 10 seconds with a stride of 9 seconds, resulting in 835,758 data points, each of size 6×10, where 6 is the stacked XYZ axes of the noised DVL and GNSS-RTK velocities. Then, the data points were split by a 80:20 [%] ratio, which resulted in 669,060 data points for the train set and 166,698 data points for the evaluation set.

To evaluate our proposed calibration approach, two error terms combinations were selected to noise the five test trajectories, T6 - T10. Table III presents the two error terms combination

selected, which are referred to as DVL1 and DVL2. Both DVL1 and DVL2 error terms were selected to mimic lower-grade DVLs.

TABLE II: DVL beam error model parameters used to create the training set.

	Lower Value	Upper Value	Step size	# Values
s_{DVL} [%]	0.2	1.5	0.1	14
b_{DVL} [cm/s]	0.1	0.9	0.1	9
σ_{DVL} [cm/s]	0.01	0.1	0.01	9

TABLE III: DVL beam error model parameters used to create the testing set. Noise values are not present in the train parameters.

	Scale [%]	Bias [cm/s]	DVL Noise [cm/s]
DVL 1	1.0	0.7	2.0
DVL 2	1.0	0.7	0.02

B. Evaluation Procedure

This section describes the evaluation procedure, which consists of calibration and evaluation phases. The procedure is designed to resemble a real-life calibration process, whereby the AUV follows a predefined calibration trajectory for a predetermined period of time, after which the error terms are estimated. Thus, the following stages were designed.

We employed the 200-second T10Cal dataset for the calibration phase. It serves a double purpose: first, the calibration error terms are estimated; secondly, the convergence time of each approach is also determined. We examined five different calibration window sizes to find the one that produces the best performance. The windows sizes are $w_i = 20, 40, 60, 80, 100$ seconds. The calibration takes place only during this window. Once the DVL calibration parameters are estimated, they are applied on the velocity measurements during the rest of the calibration phase. For example, for a 40-second window size, calibration parameters are estimated. The velocity is calibrated for the rest of the 160 seconds using the parameters estimated for the calibration during the first 40 seconds.

The accuracy of the error terms in each window for each approach is estimated by calculating the root mean squared error (RMSE) of the calibrated velocity of the remaining time against the GT, as follows:

$$\mathrm{RMSE}(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i) = \sqrt{\frac{\sum_{i=1}^{N} \sum_{j=X,Y,Z} (\boldsymbol{x}_{i,j} - \hat{\boldsymbol{x}}_{i,j})^2}{N}} \quad (29)$$

where $\boldsymbol{x}_i$ is the GT velocity vector, $\hat{\boldsymbol{x}}_i$ is the calibrated velocity vector, and N is the length of the trajectory in seconds. Once all calibration window sizes have been examined, the selected calibration parameters for each error model and the baseline approach are those that achieved the lowest RMSE (29). The corresponding window size is referred to as the calibration convergence time.

Once the calibration phase concludes, we evaluate the performance on the test dataset using the estimated calibration parameters and (29). That is, we use DCNet estimated parameters for all five error models **EM1**-**EM5**. For the baseline

approach, which calculates only the scale factor, we use (10). We examine DV1 and DVL2 parameters for the baseline and for **EM1**-**EM5**.

C. Results

To verify the accuracy of the evaluation procedure on each EM and minimize the effect of outliers, the evaluation procedure is repeated for 200 Monte Carlo (MC) iterations. For each run, the calibration parameters were evaluated in the different window sizes for each EM, and then the RMSE (29) was calculated for the calibration dataset T10Cal. The convergence time is the window size based upon the minimum RMSE that was achieved. That is:

$$\mathrm{T}_{\mathrm{conv}} = min(\mathrm{RMSE}_{20}, \ldots, \mathrm{RMSE}_{100}) \quad (30)$$

where the subscript of RMSE is the calibration window time in seconds. Note that the convergence time (30) is the required time for the calibration process. The results of the calibration phase are given in Table IV showing $\mathrm{T}_{\mathrm{conv}}$ and its associated RMSE.

TABLE IV: RMSE [cm/s] and $\mathrm{T}_{\mathrm{conv}}$ [second] for **EM1** - **EM5** using DCNet and the baseline approach during the calibration phase (T10Cal).

	Baseline	EM1	EM2	EM3	EM4	EM5
DVL1	5.94,(100)	5.95,(100)	5.94,(100)	6.29,(100)	5.96,(100)	**5.90,(40)**
DVL2	0.74,(100)	0.74,(100)	**0.73,(20)**	2.07,(100)	**0.28,(20)**	**0.2,(20)**

When considering the noising characteristics of DVL1, DCNet outperformed the baseline approach only in **EM5**, both in accuracy and in convergence time. In the latter, 60% improvement was achieved. When considering DVL2, DCNet improved on the baseline approach in accuracy and convergence time employing **EM2**, **EM4**, and **EM5**. All three EMs achieved 80% improvement in the convergence time, while **EM5** achieved the best accuracy, improving the RMSE by 72% over the baseline. Figure 8 shows the RMSE improvement at $\mathrm{T}_{\mathrm{conv}} = 20$ seconds for DCNet with **EM2**, **EM4**, and **EM5** evaluated on DVL2.
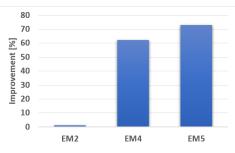

Fig. 8: RMSE at $\mathrm{T}_{\mathrm{conv}} = 20$ seconds of DCNet employing **EM2**, **EM4**, and **EM5**, evaluated on DVL2 for dataset T10Cal.

Next, we evaluate the performance of our approach and the baseline on the test dataset. To this end, we calibrate the five test trajectories—T6 - T9 and T10Test—using the estimated error terms by DCNet employing **EM1** - **EM5**, and the baseline approach, for both DVL types. The RMSE value of the calibrated test trajectories was calculated against

their GT. Table V shows the RMSE value of all evaluated approaches, and shows the improvement of DCNet over the baseline approach.

For DVL1, DCNet achieved only slightly better RMSE than the baseline approach using **EM5**, which is less than a 1% improvement. However, when considering DVL2, DCNet improved on the baseline approach by estimating error terms of **EM4** and **EM5**. Using **EM4**, DCNet improved only on test trajectories T6 - T9, which are from the same sea experiment as the training set, but not on T10Test, which is from a different sea experiment. For **EM5**, DCNet significantly improved the RMSE over the baseline approach and over all five test trajectories by an average of 70%. Figure 9 shows DCNet RMSE improvement [%] over the baseline while employing **EM5** on DVL2.

Fig. 9: Improvement [%] in RMSE of DCNet employing **EM5** over the baseline approach with DVL2.

V. CONCLUSION

This work proposes and examines the benefits of data-driven approaches in the DVL calibration process. Most works in the literature require complex trajectories accompanied by model-based nonlinear estimation filters for such a calibration task. In this work we offer a data-driven framework with a simple, yet efficient, calibration network called DCNet. We suggested five different calibration error models for the evaluation over two different types of DVL. For training the network we used a dataset of 175 minutes while the test dataset was 98 minutes long. Both datasets contain real-world recorded DVL measurements. Using **EM5**, we showed that DCNet outperforms the baseline approach by achieving an 80% improvement in calibration time. That is, instead of 100 seconds required for the calibration procedure, ours require only 20 seconds. Moreover, we demonstrate that during those 20 seconds the AUV should maintain a nearly constant velocity trajectory and does not require any complex maneuvering. In terms of the velocity accuracy, **EM5** obtained an average of 70% improvement over the baseline in terms of the velocity RMSE. In summary, this paper presents an end-to-end data-driven calibration approach that reduces the calibration process complexity while maintaining high accuracy. This holds true only for low performance (low-cost) DVL. Therefore, our work opens up the possibility of using low-cost DVLs and achieving higher accuracy. As a consequence, different types of marine robotic systems will be able to use low-cost DVLs with high accuracy opening them up for use in new applications.

REFERENCES

[1] L. Paull, S. Saeedi, M. Seto, and H. Li, "AUV navigation and localization: A review," *IEEE Journal of Oceanic Engineering*, vol. 39, no. 1, pp. 131–149, 2013.

[2] R. B. Wynn, V. A. Huvenne, T. P. Le Bas, B. J. Murton, D. P. Connelly, B. J. Bett, H. A. Ruhl, K. J. Morris, J. Peakall, D. R. Parsons *et al.*, "Autonomous Underwater Vehicles (AUVs): Their past, present and future contributions to the advancement of marine geoscience," *Marine G eology*, vol. 352, pp. 451–468, 2014.

[3] M. Jacobi, "Autonomous inspection of underwater structures," *Robotics and Autonomous Systems*, vol. 67, pp. 80–86, 2015.

[4] Y. Zhang, H. Zhang, J. Liu, S. Zhang, Z. Liu, E. Lyu, and W. Chen, "Submarine pipeline tracking technology based on AUVs with forward looking sonar," *Applied Ocean Research*, vol. 122, p. 103128, 2022.

[5] H. Zhang, S. Zhang, Y. Wang, Y. Liu, Y. Yang, T. Zhou, and H. Bian, "Subsea pipeline leak inspection by autonomous underwater vehicle," *Applied Ocean Research*, vol. 107, p. 102321, 2021.

[6] B. Li, B. Moridian, and N. Mahmoudian, "Autonomous oil spill detection: Mission planning for ASVs and AUVs with static recharging," in *OCEANS 2018 MTS/IEEE Charleston*. IEEE, 2018, pp. 1–5.

[7] A. Vasilijevic, N. Stilinovic, D. Nad, F. Mandic, N. Miskovic, and Z. Vukic, "AUV based mobile fluorometers: System for underwater oil-spill detection and quantification," in *2015 IEEE Sensors Applications Symposium (SAS)*. IEEE, 2015, pp. 1–6.

[8] J. J. Leonard and A. Bahr, "Autonomous underwater vehicle navigation," *Springer handbook of Ocean Engineering*, pp. 341–358, 2016.

[9] C. Xu, C. Xu, C. Wu, J. Liu, D. Qu, and F. Xu, "Accurate two-step filtering for AUV navigation in large deep-sea environment," *Applied Ocean Research*, vol. 115, p. 102821, 2021.

[10] X. Mu, B. He, S. Wu, X. Zhang, Y. Song, and T. Yan, "A practical INS/GPS/DVL/PS integrated navigation algorithm and its application on Autonomous Underwater Vehicle," *Applied Ocean Research*, vol. 106, p. 102441, 2021.

[11] S. Ding, T. Ma, Y. Li, S. Xu, and Z. Yang, "RD-VIO: Relative-depth-aided visual-inertial odometry for autonomous underwater vehicles," *Applied Ocean Research*, vol. 134, p. 103532, 2023.

[12] N. Cohen and I. Klein, "Seamless Underwater Navigation with Limited Doppler Velocity Log Measurements," *IEEE Transactions on Intelligent Vehicles*, pp. 1–12, 2024.

TABLE V: RMSE results of our proposed NN during the evaluation phase, that is over all the five test trajectories that were noised using the two error terms combinations. The bold font represents the lowest RMSE in each test trajectory.

RMSE [cm/s] and Improvement [%] over the baseline							
Approach / DVL type	Eval. Traj.	Baseline	DCNet				
			EM1	**EM2**	**EM3**	**EM4**	**EM5**
DVL1	T6	5.96	5.96	5.96	6.27	5.99	**5.91 ($<$1)**
	T7	5.95	5.95	5.95	6.22	5.99	**5.91 ($<$1)**
	T8	5.93	5.93	5.93	6.25	5.96	**5.89 ($<$1)**
	T9	5.95	5.95	5.95	6.26	5.99	**5.91 ($<$1)**
	T10	5.94	5.94	5.94	6.18	6.07	**5.90 ($<$1)**
DVL2	T6	0.74	0.74	0.73 (1)	1.96	0.51 (31)	**0.21 (72)**
	T7	0.74	0.74	0.74	1.78	0.57 (23)	**0.22 (70)**
	T8	0.74	0.74	0.73 (1)	1.96	0.55 (26)	**0.21 (72)**
	T9	0.74	0.74	0.73 (1)	1.91	0.63 (15)	**0.21 (72)**
	T10	0.74	0.74	0.74	1.70	1.26	**0.26 (65)**

[13] N. A. Brokloff, "Matrix algorithm for Doppler sonar navigation," in *Proceedings of OCEANS'94*, vol. 3. IEEE, 1994, pp. III–378.

[14] O. Levy and I. Klein, "INS/DVL fusion with DVL based acceleration measurements," *arXiv preprint arXiv:2308.11762*, 2023.

[15] N. Cohen, Z. Yampolsky, and I. Klein, "Set-Transformer BeamsNet for AUV velocity forecasting in complete DVL outage scenarios," in *2023 IEEE Underwater Technology (UT)*. IEEE, 2023, pp. 1–6.

[16] B. Xu and Y. Guo, "A novel DVL calibration method based on robust invariant extended Kalman filter," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 9, pp. 9422–9434, 2022.

[17] D. Wang, X. Xu, Y. Yang, and T. Zhang, "A quasi-newton quaternions calibration method for DVL error aided GNSS," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 3, pp. 2465–2477, 2021.

[18] T. Li, H. Zhang, Z. Gao, Q. Chen, and X. Niu, "High-accuracy positioning in urban environments using single-frequency multi-GNSS RTK/MEMS-IMU integration," *Remote Sensing*, vol. 10, no. 2, p. 205, 2018.

[19] Z. Ning, X. Pan, and W. Wu, "Research on fast calibration and moving base alignment of SINS/DVL integrated navigation system," *IEEE Sensors Journal*, 2023.

[20] Y. Liu, X. Fan, C. Lv, J. Wu, L. Li, and D. Ding, "An innovative information fusion method with adaptive Kalman filter for integrated INS/GPS navigation of autonomous vehicles," *Mechanical Systems and Signal Processing*, vol. 100, pp. 605–616, 2018.

[21] B. Xu, L. Wang, S. Li, and J. Zhang, "A novel calibration method of SINS/DVL integration navigation system based on quaternion," *IEEE Sensors Journal*, vol. 20, no. 16, pp. 9567–9580, 2020.

[22] J. Liu, H. Li, S. Ma, J. Luo, B. Fu, and F. Zhang, "Simultaneous calibration method for Doppler velocity log errors based on a genetic algorithm," *IEEE Sensors Journal*, vol. 22, no. 10, pp. 9558–9567, 2022.

[23] M. Yona and I. Klein, "MissBeamNet: learning missing Doppler velocity log beam measurements," *Neural Computing and Applications*, vol. 36, no. 9, pp. 4947–4958, 2024.

[24] Y. Yao, X. Xu, X. Xu, and I. Klein, "Virtual beam aided SINS/DVL tightly coupled integration method with partial DVL measurements," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 1, pp. 418–427, 2022.

[25] N. Cohen and I. Klein, "BeamsNet: A data-driven approach enhancing Doppler velocity log measurements for autonomous underwater vehicle navigation," *Engineering Applications of Artificial Intelligence*, vol. 114, p. 105216, 2022.

[26] Z. Yampolsky and I. Klein, "DVL Calibration using Data-driven Methods," *2024 IEEE/OES Oceans24 Singapore*, 2024.

[27] N. Cohen and I. Klein, "A-KIT: Adaptive Kalman-informed transformer," *arXiv preprint arXiv:2401.09987*, 2024.

[28] B. Braginsky, A. Baruch, and H. Guterman, "Correction of DVL error caused by seafloor gradient," *IEEE Sensors Journal*, vol. 20, no. 19, pp. 11 652–11 659, 2020.

[29] S. Liu, T. Zhang, and Y. Zhu, "A GNSS Aided Calibration Method for DVL Error Based on the Optimal-REQUEST," *IEEE Sensors Journal*, vol. 22, no. 22, pp. 21 899–21 910, 2022.

[30] S. Sharma, S. Sharma, and A. Athaiya, "Activation functions in neural networks," *Towards Data Sci*, vol. 6, no. 12, pp. 310–316, 2017.

[31] S. Salman and X. Liu, "Overfitting mechanism and avoidance in deep neural networks," *arXiv preprint arXiv:1901.06566*, 2019.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[33] M. C. Mukkamala and M. Hein, "Variants of RPMSProp

and Adagrad with logarithmic regret bounds," in *International Conference on Machine Learning*. PMLR, 2017, pp. 2545–2553.

[34] A. Shurin, A. Saraev, M. Yona, Y. Gutnik, S. Faber, A. Etzion, and I. Klein, "The autonomous platforms inertial dataset," *IEEE Access*, vol. 10, pp. 10 191–10 201, 2022.

[35] R. Gonzalez and C. A. Catania, "A statistical approach for optimal order adjustment of a moving average filter," in *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE, 2018, pp. 1542–1546.