A-KIT: Adaptive Kalman-Informed Transformer

Nadav Cohen and Itzik Klein
The Hatter Department of Marine Technologies, Charney School of Marine Sciences,
University of Haifa, Haifa, Israel

***Abstract*—The extended Kalman filter (EKF) is a widely adopted method for sensor fusion in navigation applications. A crucial aspect of the EKF is the online determination of the process noise covariance matrix reflecting the model uncertainty. While common EKF implementation assumes a constant process noise, in real-world scenarios, the process noise varies, leading to inaccuracies in the estimated state and potentially causing the filter to diverge. To cope with such situations, model-based adaptive EKF methods were proposed and demonstrated performance improvements, highlighting the need for a robust adaptive approach. In this paper, we derive and introduce A-KIT, an adaptive Kalman-informed transformer to learn the varying process noise covariance online. The A-KIT framework is applicable to any type of sensor fusion. Here, we present our approach to nonlinear sensor fusion based on an inertial navigation system and Doppler velocity log. By employing real recorded data from an autonomous underwater vehicle, we show that A-KIT outperforms the conventional EKF by more than 49.5% and model-based adaptive EKF by an average of 35.4% in terms of position accuracy.**

***Index Terms*—Inertial sensing, Navigation, Deep learning, Kalman filter, Sensor fusion, Estimation, Autonomous underwater vehicle**

I. Introduction

THE Kalman filter (KF) is renowned for its effectiveness in data fusion. It is acknowledged as an optimal state estimator designed for linear dynamic systems affected by Gaussian white noise [1]. In real-life applications, the majority of systems exhibit nonlinear behavior. Consequently, various extensions of the KF have been developed, such as the extended Kalman filter (EKF) [2], unscented Kalman filter (UKF) [3], and others [4]. These extensions are designed to effectively address nonlinear systems, albeit at the cost of sub-optimality. As a result, variations of the KF have found applications in diverse fields, including tracking, navigation, control, signal processing, and communication.

In the conventional KF framework, it is assumed that both process and measurement noise covariance matrices are known and well-defined and play a crucial role in conjunction with system dynamics to dictate the achievable accuracy of the filter [5, 6, 7]. However, in numerous practical scenarios, particularly when dealing with noisy feature data in nonlinear systems, the statistical properties of the noise covariances are frequently unknown or only partially discernible. Moreover, these properties may undergo changes throughout the duration of a mission [8, 9]. As a result, noise identification becomes a critical component, giving rise to adaptive filtering.

The exploration of model-based adaptive KF dates back to 1970, with Mehra [10] introducing an approach to address optimal filtering for a linear time-invariant system characterized by unknown process noise covariance. Subsequently, additional approaches emerged, categorized into four distinct groups: Bayesian inference, maximum likelihood estimation, covariance-matching, and correlation methods [11].

Recent advances in hardware and computational efficiency have demonstrated the utility of deep learning (DL) methods in addressing real-time applications. These applications span various domains, including image processing, signal processing, and natural language processing. Leveraging the inherent capabilities of DL to handle nonlinear problems, these methods have found integration into inertial navigation algorithms. Accordingly, research initiatives have been initiated to investigate data-driven estimation of the process noise covariance matrix across various domains, including land, aerial, and maritime applications [12, 13].

In the standard navigation setting, a key method for obtaining an accurate and reliable navigation solution involves the integration of measurements from various sensors through the use of a nonlinear filter, commonly an EKF. The inertial measurement unit (IMU) data is processed through the inertial navigation system (INS) equations of motion to derive a continuous, high-rate navigation solution. This system, however, is characterized by error accumulation due to the inherent nature of the inertial sensor. To mitigate this error, the EKF relies on external observations from other sensors, such as the global navigation satellite system (GNSS) or Doppler velocity log (DVL), to enhance the accuracy of the navigation solution [14, 15, 16].

In this paper, we propose a novel approach named A-KIT, an adaptive Kalman-informed transformer. Built upon a set-transformer network, A-KIT is designed for real-time adaptive regression of the process noise covariance matrix. The study showcases A-KIT's superiority, outperforming both the conventional EKF and various state-of-the-art, model-based, adaptive EKF approaches in an INS/DVL fusion case study. The key contributions of this paper include:

1) The introduction of A-KIT, a hybrid algorithm combining the strengths of the well-established EKF and leveraging well-known DL characteristics, such as noise reduction, for improved performance.
2) A novel, Kalman-informed, loss configuration designed to emulate the principles of the KF, enhancing the accuracy of the process noise covariance.
3) A GitHub repository containing experiment data and code for implementing the A-KIT architecture. The repository serves as a comprehensive resource for researchers and practitioners interested in replicating or further exploring the proposed A-KIT algorithm.

The remainder of the paper is organized as follows: In

Section II, we discuss related work. Section III delves into the theoretical and mathematical background of model-based adaptive estimation. Section IV introduces the components of the set-transformer architecture. In Section V, we present the proposed A-KIT approach. Section VI establishes the case study of the INS/DVL sensor fusion and discusses the implementation and integration of A-KIT into this problem. In Section VII, the dataset acquisition process is detailed, along with an in-depth analysis of the results. Finally, Section VIII discusses the conclusions.

II. Related Work

When examining model-based adaptive KF approaches, various studies have demonstrated improvements in INS and GNSS navigation across different platforms using innovation-based adaptive KF [17, 18, 19, 20]. Additionally, the effectiveness of adaptive KFs in INS/DVL applications has been showcased in works such as [21, 22, 23, 24, 25].

For land-based navigation in the context of terrestrial vehicles, Brossard, Barrau, and Bonnabel proposed a method that utilizes convolutional neural networks (CNNs) to dynamically adjust the covariance noise matrix within an invariant EKF. This adaptation is achieved by incorporating moderately priced IMU measurements [26]. In an alternative approach, specifically designed for land vehicles, reinforcement learning takes center stage, as elucidated in [27]. Wu *et al.* [28] conducted a study where not only were the parameters of measurement noise covariances regressed, but also the parameters of process noise covariances were predicted. This prediction utilized a multitask temporal convolutional network (TCN), resulting in superior position accuracy compared to traditional GNSS/INS integrated navigation systems. In a unique approach, Xiao *et al.* [29] introduced a residual network incorporating an attention mechanism specifically designed to predict individual velocity elements of the noise covariance matrix, with a focus on land vehicle applications.

For aerial platforms, a CNN-based adaptive Kalman filter was proposed by Zou *et al.* [30] to enhance high-speed navigation using low-cost IMU. The study introduces a 1D CNN approach to predict 3D acceleration and angular velocity noise covariance information. In subsequent work, Or and Klein [31] introduced a data-driven, adaptive noise covariance approach for an error state EKF in INS/GNSS fusion. The obtained information was utilized to enhance the navigation solution for a quadrotor drone.

When examining autonomous underwater vehicle (AUV) navigation with INS/DVL sensor fusion, Huang *et al.* [32] proposed the utilization of a long short-term memory (LSTM) network to estimate the initial and constant process noise covariance matrix. Their work demonstrated significant improvement over conventional EKF approaches. Or and Klein [25, 31, 33] devised an adaptive EKF customized for velocity updates in INS/DVL fusions. Initially, they highlighted the efficacy of correcting the noise covariance matrix through a 1D CNN, utilizing a classification approach to predict the variance at each sample time. This correction method resulted in substantial improvements in navigation results. In their most recent work, the authors introduced a novel approach that employs regression instead of classification to achieve the same objective. However, a crucial observation must be noted, that while the approach demonstrated success when tested on simulated data, its performance did not translate effectively when applied to real-world data obtained from an AUV.

In the exploration of related work, Haarnoja *et al.* [34] were pioneers in examining a closed-loop KF solution. They approached the problem by drawing parallels to recurrent neural networks (RNNs) and highlighted the necessity of implementing a backpropagation through time approach, emphasizing its computational cost. Their methodology was tested for solving the visual odometry problem, and the designed 1D CNN provided the update measurement and the measurement noise covariance. In [35], the authors proposed an RNN to dynamically adapt the process noise of a constant velocity KF in a tracking problem. However, it is worth noting that the methods presented in the above-mentioned works were not directly compared to model-based adaptive KF approaches.

III. Model-Based Adaptive Estimation

A prevalent method for conducting state estimation with a given model and set of observations is the KF. In accordance with the theory, the filter leads to the optimal linear state estimator that minimizes the mean square error under Gaussian assumptions [1, 8]. However, assuming a linear model in real-world applications such as navigation and control is unrealistic, as these scenarios typically involve nonlinear problems. To address this, an extension of the linear Kalman Filter, known as the EKF, capable of handling nonlinear problems, was introduced [36]. In this section, a concise theoretical background is presented, introducing both the error-state EKF and the adaptive error-state EKF.

A. Conventional Extended Kalman Filter

We describe the error-state implementation of the EKF (ES-EKF). To that end, consider $\boldsymbol{\delta x} \in \mathbb{R}^{n\times 1}$, an n-dimensional error-state vector, defined as follows:

$$\boldsymbol{x}^t = \boldsymbol{x}^e - \boldsymbol{\delta x} \tag{1}$$

where $\boldsymbol{x}^t \in \mathbb{R}^{n\times 1}$ and $\boldsymbol{x}^e \in \mathbb{R}^{n\times 1}$ are the true state and the estimated state, respectively. In the EKF, the underlying assumption is that the error in the state vector estimate is substantially smaller than the state vector. This condition facilitates the application of a linear system model to the state vector residual:

$$\boldsymbol{\delta \dot{x}} = \mathbf{F}\boldsymbol{\delta x} + \mathbf{G}\boldsymbol{n} \tag{2}$$

where $\boldsymbol{n} \in \mathbb{R}^{n\times 1}$ is the system noise vector, $\mathbf{F} \in \mathbb{R}^{n\times n}$ is the system matrix, and $\mathbf{G} \in \mathbb{R}^{n\times n}$ is the system noise distribution (shaping) matrix. The Kalman filtering process typically involves two distinct phases: the prediction and update steps.

In the prediction step, the *a priori* error state, $\delta\boldsymbol{x}^-$, is considered equal to zero to facilitate linearization, a crucial aspect of the EKF method. The state covariance is then propagated using the known model:

$$\delta\boldsymbol{x}^- = 0 \tag{3}$$

$$\mathbf{P}_k^- = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1}^+\mathbf{\Phi}_{k-1}^T + \mathbf{Q}_{k-1} \tag{4}$$

where $\mathbf{P}_k^-$ is the *a priori* state covariance estimate at time k and $\mathbf{P}_{k-1}^+$ is the *a posteriori* state covariance estimate at time $k-1$. The transition matrix, denoted as $\mathbf{\Phi}_{k-1}$, is typically derived through a power-series expansion of the system matrix, $\mathbf{F}$, and the propagation interval, τ_s:

$$\mathbf{\Phi}_{k-1} = \sum_{r=0}^{\infty} \frac{\mathbf{F}_{k-1}^r}{r!}\tau_s^r \tag{5}$$

and $\mathbf{F}^{k-1}$ is defined as:

$$\mathbf{F}^{k-1} = \frac{\partial \boldsymbol{f}(\boldsymbol{x})}{\partial \boldsymbol{x}}|_{x=x^e} \tag{6}$$

where $\boldsymbol{f}(\boldsymbol{x})$ is the nonlinear function of the state propagation. The exact form of the system noise covariance matrix is [36]:

$$\mathbf{Q}_{k-1} = \mathbb{E}\Big(\int_{t-\tau_s}^{t}\int_{t-\tau_s}^{t} \mathbf{\Phi}_{k-1}(t-t')\mathbf{G}_{k-1}\boldsymbol{n}(t') \boldsymbol{n}^T(t'')\mathbf{G}_{k-1}^T\mathbf{\Phi}_{k-1}^T(t-t'')\,\partial t'\,\partial t''\Big) \tag{7}$$

which is usually approximated and can be written as follows [37]:

$$\mathbf{Q}_{k-1} = \frac{1}{2}\left(\mathbf{\Phi}_{k-1}\mathbf{G}_{k-1}\mathbf{Q}\mathbf{G}_{k-1}^T + \mathbf{G}_{k-1}\mathbf{Q}\mathbf{G}_{k-1}^T\mathbf{\Phi}_{k-1}^T\right)\Delta t. \tag{8}$$

The next phase is the update step, which is executed by the following equations:

$$\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}_k^T\left(\mathbf{H}_k\mathbf{P}_k^-\mathbf{H}_k^T + \mathbf{R}_k\right)^{-1} \tag{9}$$

$$\mathbf{P}_k^+ = [\mathbf{I} - \mathbf{K}_k\mathbf{H}_k]\mathbf{P}_k^- \tag{10}$$

$$\boldsymbol{\delta x}_k^+ = \mathbf{K}_k\boldsymbol{\delta z}_k \tag{11}$$

where $\mathbf{K}_k$ is the Kalman gain, dictating the balance between incorporating new measurements and the predictions generated by the system's dynamic model. The $\mathbf{H}_k$ and $\mathbf{R}_k$ matrices correspond to the measurement matrix and the covariance matrix of measurement noise, respectively. Lastly, $\boldsymbol{\delta x}_k^+$ and $\mathbf{P}_k^+$ are the *a posteriori* error state and estimated covariance state, respectively. The measurement innovation, $\boldsymbol{\delta z}_k$, is the residual between the state given by the model and the measurement provided by the observation. The measurement matrix, $\mathbf{H}_k$, is defined by:

$$\mathbf{H}_k = \frac{\partial \boldsymbol{h}(\boldsymbol{x})}{\partial \boldsymbol{x}}|_{x=x^e} \tag{12}$$

where $\boldsymbol{h}(\boldsymbol{x})$ represents the update equations characterized by nonlinear functions of the state vector.

B. Adaptive Error-State Extended Kalman Filter

The conventional ES-EKF assumes the noise covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ to be constant throughout the estimation process. However, it has been shown in the literature that when the measurement noise covariances are unknown or varying in time, the adaptive approach outperforms the conventional one [17]. Specifically, we look at the innovation-based adaptive estimation (IAE) approach window, commonly used in different applications [38]. The covariance matrix $\mathbf{Q}_k$ is adjusted as measurements evolve over time. This adaptation of the filter's statistical information matrix is guided by the whiteness of the filter's innovation sequence and takes the following form:

$$\hat{\mathbf{Q}}_k = \mathbf{K}_k\hat{\mathbf{C}}_{v_k}\mathbf{K}_k^T. \tag{13}$$

To calculate $\hat{\mathbf{C}}_{v_k}$, we perform averaging within a moving estimation window of size N. This process is carried out as follows:

$$\hat{\mathbf{C}}_{v_k} = \frac{1}{N}\sum_{j=j_0}^{k}\boldsymbol{\delta z}_j\boldsymbol{\delta z}_j^T \tag{14}$$

where $j_0 = k - N + 1$ is the first iteration of the process noise covariance estimation. Equation (13) is used to replace the constant process evaluation (8) inside the conventional ES-EKF.

The approach outlined above represents the fundamental adaptive method. Over time, the literature has introduced various alternative approaches. In this context, we elucidate two of these variations to provide a comparative foundation for the suggested approach. The first approach is described in [39, 40] and seeks to scale to process noise covariance. The scaling factor is defined as:

$$\beta = \frac{\mathrm{Tr}\{\mathbf{H}_k\left(\mathbf{\Phi}_{k-1}\mathbf{P}_{k-1}^+\mathbf{\Phi}_{k-1}^T + \hat{\mathbf{Q}}_{k-1}\right)\mathbf{H}_k^T\}}{\mathrm{Tr}\{\mathbf{H}_k\left(\mathbf{\Phi}_{k-1}\mathbf{P}_{k-1}^+\mathbf{\Phi}_{k-1}^T + \mathbf{Q}_{k-1}\right)\mathbf{H}_k^T\}} \tag{15}$$

and the adaptation rule is defined as:

$$\hat{\mathbf{Q}}_k = \hat{\mathbf{Q}}_{k-1}\sqrt{\beta}. \tag{16}$$

The third adaptive approach utilizes a forgetting factor γ to average estimates of $\mathbf{Q}_k$ over time [41]:

$$\hat{\mathbf{Q}}_k = \gamma\mathbf{Q}_{k-1} + (1-\gamma)\hat{\mathbf{Q}}_{k-1} \tag{17}$$

where the manually tuned factor γ satisfies $0 \leq \gamma \leq 1$. A larger value of γ places more weight on previous estimates, incurring less fluctuation in the current estimate but resulting in longer time delays to catch up with changes.

IV. Set-Transformer

This section reviews the building blocks of the set-transformer designed to handle time-series data.

A. Patch Embedding

Following the approach recommended by Dosovitskiy *et al.* [42], the data undergoes a preprocessing step wherein it is partitioned into patches, flattened, and subsequently subjected to a one-dimensional convolution operation. This convolution operation employs a kernel size denoted as α and a stride size as β, and utilizes patch sizes represented by γ. The outcome of this patch embedding process is denoted as $\mathbf{x}_p$ and resides in $\mathbb{R}^{N\times D}$, where N signifies the quantity of samples produced by the one-dimensional convolution layer, and D corresponds to the number of filters employed by the layer and also establishes the latent space dimension that is subsequently input into the set-transformer blocks.

B. Attention

The attention mechanism is formally defined by the equation

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_q}}\right)\mathbf{V} \in \mathbb{R}^{n \times d_v}. \tag{18}$$

Here, $\mathbf{Q}$ represents n query vectors of size d_q in $\mathbb{R}^{n \times d_q}$, and $\mathbf{K}$ and $\mathbf{V}$ denote n_v key-value pairs in $\mathbb{R}^{n_v \times d_q}$ and $\mathbb{R}^{n_v \times d_v}$, respectively. The dimension of the values vector is denoted as d_v.

The Softmax function, as described in [43], is expressed as

$$\text{Softmax}(\boldsymbol{Z})i = \frac{e^{Zi}}{\sum_{n=1}^{M} e^{Z_j}}. \tag{19}$$

In this equation, M denotes the number of vectors.

C. Multihead Attention

Multiheads are introduced to extend the capabilities of the attention mechanism. In this model, query vectors ($\mathbf{Q}$), key vectors ($\mathbf{K}$), and value vectors ($\mathbf{V}$) are projected onto h different heads, each having dimensions $\acute{d_q}$, $\acute{d_q}$, and $\acute{d_v}$, respectively. The previously described attention mechanism in (18) is then independently applied to each of these h projections.

This multihead attention mechanism allows the model to effectively incorporate information from various representation subspaces and positions, enhancing its capacity to capture diverse patterns in the data.

Within this model, trainable parameters $\mathbf{W}_j^Q, \mathbf{W}_j^K \in \mathbb{R}^{d_q \times \acute{d_q}}$, and $\mathbf{W}_j^V \in \mathbb{R}^{d_v \times \acute{d_v}}$ are employed in the following manner:

$$head_j = \text{Attention}(\mathbf{Q}\mathbf{W}_j^Q, \mathbf{K}\mathbf{W}_j^K, \mathbf{V}\mathbf{W}_j^V),\ j = 1, ..., h. \tag{20}$$

These multiple heads are then linked together in a sequential manner and are scaled by a trainable parameter $\mathbf{W}^O \in \mathbb{R}^{h\acute{d_v} \times d}$, resulting in the following expression:

$$\text{Multihead}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{concatenate}(head_1, ..., head_h)\mathbf{W}^O. \tag{21}$$

A common choice for the hyperparameters in terms of dimensions is $\acute{d_q} = \frac{d_q}{h}$, $\acute{d_v} = \frac{d_v}{h}$, and $d = d_q$. These choices help manage the dimensionality of the multihead attention mechanism effectively.

D. Set Attention Block

The multihead attention block (MAB) represents a pivotal modification to the core encoder block of the transformer architecture, as originally introduced by Vaswani *et al.* [44]. This adaptation, characterized by the removal of positional encoding and dropout operations, is designed to harness the intrinsic power of self-attention mechanisms. In the context of MAB, two matrices, $\mathbf{X}$ and $\mathbf{Y}$, both residing in $\mathbb{R}^{n \times d}$, play a central role. The parameter $\mathbf{L}$ is defined as follows:

$$\mathbf{L} = \text{LayerNorm}(\mathbf{X} + \text{Multihead}(\mathbf{X}, \mathbf{Y}, \mathbf{Y})). \tag{22}$$

The MAB is an integral component of our architecture, defined as

$$\text{MAB}(\mathbf{X}, \mathbf{Y}) = \text{LayerNorm}(\mathbf{L} + \text{FFN}(\mathbf{L})). \tag{23}$$

In this context, FFN denotes a fully connected, feed-forward network. Specifically, it comprises a layer that expands the dimensionality to the size specified by the hyper-parameter ffe, which corresponds to the feed-forward expansion. Following the application of a rectified linear unit (ReLU) activation function, the second layer restores the data to its original dimension:

$$\text{FFN}(\mathbf{X}) = \max(0, \mathbf{X}W_1 + b_1)W_2 + b_2 \tag{24}$$

where W_j and b_j are the weights and biases, respectively, such that $j \in \{1, 2\}$.

The set attention block (SAB) assumes the crucial role of performing self-attention operations among elements within the set. Importantly, it can be succinctly defined by invoking the MAB:

$$SAB(\mathbf{X}) := \text{MAB}(\mathbf{X}, \mathbf{X}). \tag{25}$$

This architectural adaptation facilitates a holistic understanding of relationships among set elements, significantly enhancing the model's representational capacity.

E. Pooling by Multihead Attention

This block introduces a learnable aggregation mechanism by establishing a set of k vectors, each with a size of n, which are subsequently organized into a matrix denoted as $\mathbf{S} \in \mathbb{R}^{k \times n}$. In this context, assuming that an encoder generates a set of features represented as $\mathbf{Z} \in \mathbb{R}^{n \times d}$, the pooling by multihead attention block (PMA) is defined as follows:

$$PMA_k(\mathbf{Z}) = MAB(\mathbf{S}, \text{FFN}(\mathbf{Z})) \tag{26}$$

where k represents a tunable hyper-parameter, integral to the functionality of this block.

F. Overall Architecture

The architectural framework comprises an encoder followed by a decoder. Given an input matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$, the encoder generates the feature matrix $\mathbf{Z} \in \mathbb{R}^{n \times d}$. The encoder's composition involves the stacking of set attention block (SAB) modules, as indicated in (25):

$$\mathbf{Z} = \text{Encoder}(\mathbf{X}) = SAB_1 \circ ... \circ SAB_b(\mathbf{X}). \tag{27}$$

Subsequently, the decoder utilizes a learnable aggregation scheme to transform $\mathbf{Z}$ into a set of vectors that traverse a feed-forward network to produce the final outputs [45]:

$$\text{Decoder}(\mathbf{Z}) = \circ \ldots \circ \text{FFN}_b(\text{SAB}(PMA_k(\mathbf{Z}))) \in \mathbb{R}^{k \times d} \quad (28)$$

where b signifies the number of stacked blocks.
In summary, the critical hyperparameters that require specification within these blocks encompass:

- α - Kernel size for patch embedding
- β - Stride size for patch embedding
- γ - Patch size for patch embedding
- D - Number of filters in the 1D convolution layer
- d - Latent space dimension, equating to D
- h - Number of attention heads
- ffe - Feed-forward expansion
- b - Number of stacked SAB blocks
- k - Number of trainable vectors for aggregation

It is noteworthy that the original set-transformer, initially designed for natural language processing to handle vectorized words within an injective vocabulary, has been adapted for one-dimensional data characterized by infinite values. This adaptation incorporates the patch embedding technique, originally proposed for vision transformers. Notably, in this context, a 1D-CNN is employed in lieu of a 2D-CNN.

V. Proposed Approach

To cope with real-time adaptive process noise covariance matrix estimation, we propose A-KIT, an adaptive Kalman-informed transformer. To this end, we derive a tailored set-transformer network for time series data dedicated to real-time regression of the EKF's process noise covariance matrix. Additionally, a Kalman-informed loss is designed to emulate the principles of the KF, enhancing the accuracy of the process noise covariance. In this manner, A-KIT is designed as a hybrid algorithm combining the strengths of the well-established theory behind EKF and leveraging well-known, deep-learning characteristics.

The A-KIT cycle begins with the initialization of the EKF, followed by the execution of the prediction phase in a loop until a valid observation is obtained. Throughout the prediction step, the estimated covariance $\mathbf{P}_k$ is stored in memory, along with all the transition matrices $\boldsymbol{\Phi}_k$, up until the next update step. Upon encountering a valid observation, the update step is initiated, calculating the innovation-based, process noise covariance. Simultaneously, the A-KIT network receives the inertial measurement and estimated state $\boldsymbol{x}^n_{EKF}$ from the filter. It then computes the scaling factor necessary to multiply the innovation-based process noise covariance

$$\mathbf{Q}_k = \hat{\mathbf{Q}}_k \cdot \mathbf{Q}_k^{A-KIT} \quad (29)$$

where $\mathbf{Q}_k^{A-KIT}$ is a diagonal scale factor matrix produced by A-KIT. The Kalman-informed loss, described in section V-A1, utilizes the pre-saved estimated covariance matrix and transition matrices to perform the prediction step once again with the output from the A-KIT. Subsequently, it calculates the Kalman gain to correct the error state. The entire A-KIT flowchart is depicted in Fig. 1. As A-KIT is intended to regress the process noise, both prediction and update stages of the EKF are present in the cycle, allowing the propagation of the error-state and its associated error-state covariance.

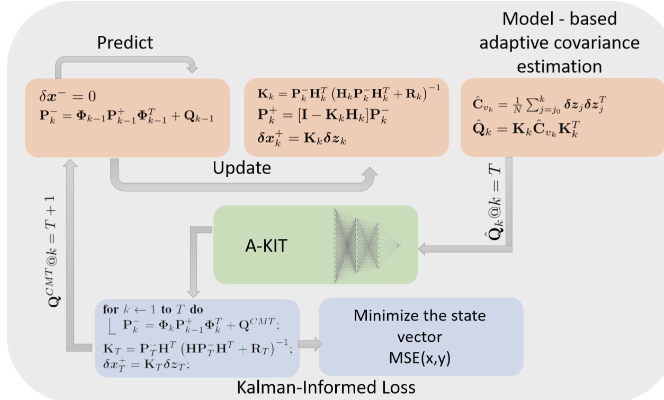


Fig. 1: Flowchart of the A-KIT, illustrating the dynamic estimation of the process noise covariance.

A. A-KIT Algorithm

1) Kalman-Informed Loss

To align the A-KIT network with the model-based EKF, we propose a Kalman-informed loss. To that end, besides the network output, the loss incorporates several EKF parameters. The first is the estimated *a posteriori* state covariance matrix $\mathbf{P}_{k-1}^+$, obtained immediately after a Kalman update. The second set of parameters includes the transition matrices $\boldsymbol{\Phi}_k$ between updates for model propagation, the measurement matrix $\mathbf{H}_k$ at the time of the update step, the *a priori* state estimate, the measurement innovation $\boldsymbol{\delta z}_k$, the covariance matrix of measurement noise $\mathbf{R}_k$, and the labels, which represent the ground-truth (GT) state.

The loss mirrors the stages of the KF, with the process noise covariance serving as the output of the network, optimized through back-propagation. In essence, the loss function addresses the question: given that sensor fusion is a nonlinear problem, what is the optimal process noise covariance matrix that minimizes the mean squared error of the measured state? The choice to focus on the measured state rather than other navigation parameters stems from its status as an observable state. The algorithm for the Kalman-informed loss is outlined in Algorithm 1.

Algorithm 1: Kalman-informed loss

Input: $\mathbf{Q}^{A-KIT}$, $\mathbf{P}_{k-1}^+$, $\boldsymbol{\Phi}_k$, $\mathbf{H}_k$, $\boldsymbol{x}^n_{EKF}$, $\boldsymbol{\delta z}_k$, $\mathbf{R}_k$, $\boldsymbol{x}^n_{GT}$

for $k \leftarrow 1$ **to** T **do**

$\quad \mathbf{P}_k^- = \boldsymbol{\Phi}_k \mathbf{P}_{k-1}^+ \boldsymbol{\Phi}_k^T + \mathbf{Q}^{A-KIT}$;

$\mathbf{K}_T = \mathbf{P}_T^- \mathbf{H}_T^T \left(\mathbf{H}_T \mathbf{P}_T^- \mathbf{H}_T^T + \mathbf{R}_T\right)^{-1}$;

$\boldsymbol{\delta x}_T^+ = \mathbf{K}_T \boldsymbol{\delta z}_T$;

$\boldsymbol{x}^n_{A-KIT} = \boldsymbol{x}^n_{EKF} - \boldsymbol{\delta x}_T^+$;

$Loss = MSE\{\boldsymbol{x}^n_{A-KIT}, \boldsymbol{x}^n_{GT}\}$;

Output: $Loss$

---

2) A-KIT Algorithm

Once the network is trained through the aforementioned stages, it can be seamlessly integrated into the EKF methodology before the prediction step to obtain an accurate process noise covariance matrix. The algorithm is detailed in Algorithm 2.

Algorithm 2: A-KIT over EKF

Input: $\hat{\boldsymbol{f}}^b_{ib}$, $\hat{\boldsymbol{\omega}}^b_{ib}$, $\boldsymbol{z^b}$, Innovation window size, $\mathbf{Q}$, $\mathbf{R}$

Initialize: $\boldsymbol{\delta x^n_s}$, $\boldsymbol{\epsilon^n}$, $\boldsymbol{\delta b_a}$, $\boldsymbol{\delta b_g}$, $\mathbf{P}$

for $k \leftarrow 1$ **to** T **do**

- $\hat{\boldsymbol{f}}^b_{ib} \leftarrow \hat{\boldsymbol{f}}^b_{ib} - \boldsymbol{\delta b_a}$;
- $\hat{\boldsymbol{\omega}}^b_{ib} \leftarrow \hat{\boldsymbol{\omega}}^b_{ib} - \boldsymbol{\delta b_g}$;
- Calculate $\mathbf{F}$ (6);
- Obtain $\boldsymbol{\Phi}_{k-1}$ (5);
- Calculate $\mathbf{G}$ (2);
- Perform the prediction step (4);
- Solve INS ;
- **if** *update is valid* **then**
  - Get $\boldsymbol{\delta z}$;
  - Calculate $\mathbf{H}_k$ (12);
  - Perform the update step (9)-(11);
  - Correct the INS measurements ;
  - **if** *Num. of updates* $\geq$ *Innovation window size* **then**
    - $\hat{\mathbf{Q}}_k \leftarrow$ innovation covariance (13)-(14) ;
    - $\mathbf{Q}^{A-KIT}_k \leftarrow$ A-KIT($\hat{\boldsymbol{f}}^b_{ib}$, $\hat{\boldsymbol{\omega}}^b_{ib}$,$\hat{\boldsymbol{x}}^n_{EKF}$,$\hat{\mathbf{Q}}_k$);
    - $\mathbf{Q}_k \leftarrow \hat{\mathbf{Q}}_k \cdot \mathbf{Q}^{A-KIT}_k$(29);

Output: $\boldsymbol{x^n}$

Here, $\boldsymbol{x}^n$ is the estimated state, $\boldsymbol{\delta x}^n_s$ is a subset of the error state—which could be the position or velocity pending on the external measurement type—and $\boldsymbol{z}^b$ is the measurement update. A-KIT provides an estimate for the process noise contrivance, enabling the propagation of both error-state and its associated error-state covariance.

VI. Case Study: INS/DVL Fusion

Developing a reliable autonomous navigation system is essential for navigating deep underwater environments, particularly in areas beyond human reach. Commonly, underwater navigation relies on the fusion between INS and DVL in an EKF framework, with velocity corrections from the DVL to ensure accurate navigation. The subsequent exploration into navigation components unfolds in the following sections: VI-A reference frames, establishing foundational spatial context; VI-B dead-reckoning, a method vital for a continuous position, velocity, and orientation deduction; VI-C DVL velocity estimation, highlighting the role of Doppler velocity logs in enhancing accuracy; and VI-D conventional EKF and III-B adaptive EKF, illuminating advanced filtering strategies essential for refining navigation solutions in complex underwater scenarios.

A. Reference Frames

Three reference frames are addressed in this work:

- **Body frame:** The body center is located at the center of mass of the vehicle. The x-axis is set along the vehicle's longitudinal axis, pointing forward, the z-axis downward, and the y-axis pointed outward, completing the right-hand orthogonal coordinate system.
- **Navigation frame:** The navigation or geographic frame is defined locally, relative to the Earth's geoid. We employ the north-east-down (NED) coordinate system where the x-axis points to the true north, the y-axis points east, and the z-axis points toward the interior of the ellipsoid along the ellipsoid normal, the direction of gravity, completing a right-hand orthogonal coordinate system
- **DVL frame:** The DVL frame is a reference frame for the DVL sensor, whose sensitive axes are designed by the manufacturer. The transformation from the DVL frame to the body frame is represented by a known fixed transformation matrix. [46].

B. Dead Reckoning

Dead reckoning (DR) is the basic component of underwater navigation. Since GNSS signals do not penetrate water and cannot be used to aid navigation, relying on initial conditions and integrating inertial data over time is necessary. In a terrestrial navigation system, which operates in the local geographic reference frame, the time derivative of the coordinates can be written as

$$\dot{\boldsymbol{r}}^n = \begin{bmatrix} \dot{\varphi} \\ \dot{\lambda} \\ \dot{h} \end{bmatrix} = \begin{bmatrix} \frac{1}{M+h} & 0 & 0 \\ 0 & \frac{1}{(N+h)\cos\varphi} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} v_N \\ v_E \\ v_D \end{bmatrix} = \mathbf{D}\boldsymbol{v}^n \quad (30)$$

where $\boldsymbol{r}^n$ represents the position in the geographic frame and φ, λ, and h are the latitude, longitude, and height, respectively, and M and N are radii of curvature in the meridian and prime vertical [47]. The velocity dynamics is expressed by

$$\dot{\boldsymbol{v}}^n = \mathbf{C}^n_b \boldsymbol{f}^b_{ib} - (2\boldsymbol{\omega}^n_{ie} + \boldsymbol{\omega}^n_{en}) \times \boldsymbol{v}^n + \boldsymbol{g}^n \quad (31)$$

where $\boldsymbol{v}^n$ is the velocity vector expressed in the navigation frame, $\boldsymbol{f}^b_{ib}$ is the specific force vector expressed in the body frame, $\boldsymbol{g}^n$ is the local gravity vector expressed in the navigation frame, and ω_e is the magnitude of the rotation rate of the Earth and has the value of 7.29×10^{-5} $\frac{rad}{Sec}$. The vectors $\boldsymbol{\omega}^n_{ie}$ and $\boldsymbol{\omega}^n_{en}$ are defined as

$$\boldsymbol{\omega}^n_{ie} = [\omega_e \cos\varphi \quad 0 \quad -\omega_e \sin\varphi]^T \quad (32)$$

$$\boldsymbol{\omega}^n_{en} = [\dot{\lambda}\cos\varphi \quad -\dot{\varphi} \quad -\dot{\lambda}\sin\varphi]^T. \quad (33)$$

The rate of change of the rotation matrix from body to navigation frame is

$$\dot{\mathbf{C}}^n_b = \mathbf{C}^n_b \boldsymbol{\Omega}^b_{ib} - (\boldsymbol{\Omega}^n_{ie} + \boldsymbol{\Omega}^n_{en})\mathbf{C}^n_b. \quad (34)$$

The body to navigation transformation matrix $\mathbf{C}^n_b$ is defined as

$$\mathbf{C}^n_b = \begin{bmatrix} c\theta c\psi & -c\phi s\psi + s\phi s\theta c\psi & s\phi s\psi + c\phi s\theta c\psi \\ c\theta c\psi & c\phi s\psi + s\phi s\theta c\psi & -s\phi s\psi + c\phi s\theta c\psi \\ -s\theta & s\phi c\theta & c\phi c\theta \end{bmatrix} \quad (35)$$

where sin and cos are denoted as s and c, respectively. ϕ, θ, and ψ are the three components of the Euler angles roll, pitch, and yaw, respectively. The notation of Ω_a^b is a skew-symmetric form of a vector defined as follows:

$$\Omega_a^b = (\omega_a^b \times) = ([\omega_1 \quad \omega_2 \quad \omega_3]\times) = \\ = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{36}$$

and operates on $\boldsymbol{\omega}_{ie}^n$ and $\boldsymbol{\omega}_{en}^n$, which are defined in (32) - (33). Additionally, the operator is used on $\boldsymbol{\omega}_{ib}^b$, which is the output of the gyroscope sensor used in (34) [15].

A navigation solution can be established by using equations (30) - (35). Since sensor data is subject to errors such as stochastic noise and bias, the inertial measurements are expressed as follows:

$$\begin{aligned} \hat{\boldsymbol{f}}_{ib}^b &= \boldsymbol{f}_{ib}^b + \boldsymbol{b}_f + \boldsymbol{n}_f \\ \hat{\boldsymbol{\omega}}_{ib}^b &= \boldsymbol{\omega}_{ib}^b + \boldsymbol{b}_\omega + \boldsymbol{n}_\omega \end{aligned} \tag{37}$$

such that the accelerometer and gyro noises are defined as zero mean white Gaussian noise:

$$\boldsymbol{n}_a \sim \mathcal{N}(0, \sigma_a^2), \; \boldsymbol{n}_g \sim \mathcal{N}(0, \sigma_g^2) \tag{38}$$

and the biases are assumed to be modeled using random walk processes

$$\begin{aligned} \dot{\boldsymbol{b}}_a &= \boldsymbol{n}_{a_b}, \; \boldsymbol{n}_{a_b} \sim \mathcal{N}(0, \sigma_{a_b}^2) \\ \dot{\boldsymbol{b}}_g &= \boldsymbol{n}_{g_b}, \; \boldsymbol{n}_{g_b} \sim \mathcal{N}(0, \sigma_{g_b}^2) \end{aligned} \tag{39}$$

where $\boldsymbol{b}_a$ and $\boldsymbol{b}_g$ are the biases of the accelerometer and gyroscope, respectively. As the inertial readings (37) are integrated, the navigation solution accumulates error with time. Therefore, the INS is fused with additional sensors [36].

C. DVL Velocity Estimation

The DVL is an acoustic-based sensor that transmits four acoustic beams to the seabed in an $\times$ shape configuration, also known as the "Janus Doppler configuration". The sensor acts as both the transmitter and the receiver and once the acoustic beams are reflected back, due to the Doppler effect, the velocity can be estimated [48, 49]. In fact, the raw measurements are the velocity in the beam directions and can be expressed in the following manner:

$$\boldsymbol{v}_{beam} = \mathbf{H}_{beam}\boldsymbol{v}^d, \; \mathbf{H}_{beam} = \begin{bmatrix} c\psi_{beam_1}s\alpha & s\psi_{beam_1}s\alpha & c\alpha \\ c\psi_{beam_2}s\alpha & s\psi_{beam_2}s\alpha & c\alpha \\ c\psi_{beam_3}s\alpha & s\psi_{beam_3}s\alpha & c\alpha \\ c\psi_{beam_4}s\alpha & s\psi_{beam_4}s\alpha & c\alpha \end{bmatrix} \tag{40}$$

where sin and cos are denoted as s and c, respectively, $\boldsymbol{v}_{beam} \in \mathbb{R}^{4\times1}$ is the velocity in the beam directions, and $\boldsymbol{v}^d \in \mathbb{R}^{3\times1}$ is the velocity in the DVL frame. Each beam transactor is rotated by a yaw angle ψ_{beam} and a pitch angle α, such that α is constant and predefined by the manufacturer, and ψ_{beam} is expressed as follows:

$$\psi_{beam_i} = (i-1)\cdot 90^\circ + 45^\circ \;, \; i = 1,2,3,4. \tag{41}$$

The measurements are not obtained ideally as illustrated in (40) and are subject to built-in errors that are modeled by:

$$\boldsymbol{y} = \mathbf{H}_{beam}[\boldsymbol{v}^d(\mathbf{1} + \boldsymbol{s}_{DVL})] + \boldsymbol{b}_{DVL} + \boldsymbol{n} \tag{42}$$

where $\boldsymbol{b}_{DVL} \in \mathbb{R}^{4\times1}$ is the bias vector, $\boldsymbol{s}_{DVL} \in \mathbb{R}^{3\times1}$ is the scale factor vector, and $\boldsymbol{n} \in \mathbb{R}^{4\times1}$ is a zero mean white Gaussian noise [50, 51].

Once the raw measurements are obtained, the next phase is to extract $\boldsymbol{v}^d$ by filtering the data according to the following cost function:

$$\hat{\boldsymbol{v}}^d = \underset{\boldsymbol{v}^d}{\operatorname{argmin}} || \, \boldsymbol{y} - \mathbf{H}_{beam}\boldsymbol{v}^d \, ||^2. \tag{43}$$

The solution to (43) is [1]:

$$\hat{\boldsymbol{v}}^d = (\mathbf{H}_{beam}^T\mathbf{H}_{beam})^{-1}\mathbf{H}_{beam}^T\boldsymbol{y}. \tag{44}$$

Finally, the DVL's estimated velocity vector is transformed to the body frame using [52]:

$$\hat{\boldsymbol{v}}^b = \mathbf{C}_d^b\hat{\boldsymbol{v}}^d \tag{45}$$

where $\mathbf{C}_d^b$ is the DVL to body transformation matrix and $\hat{\boldsymbol{v}}^b$ is the DVL velocity in the body frame.

D. INS/DVL Fusion

In the case of DVL velocity updates, the position is not directly observable [53]; hence, a twelve-state vector residual, $\boldsymbol{\delta x^n}$, is employed:

$$\boldsymbol{\delta x^n}^T = [\boldsymbol{\delta v^n}^T \quad \boldsymbol{\epsilon^n}^T \quad \boldsymbol{\delta b_a}^T \quad \boldsymbol{\delta b_g}^T]^T \in \mathbb{R}^{12\times1} \tag{46}$$

where $\boldsymbol{\delta v^n} \in \mathbb{R}^{3\times1}, \boldsymbol{\delta\epsilon} \in \mathbb{R}^{3\times1}, \boldsymbol{b_a} \in \mathbb{R}^{3\times1}$ and $\boldsymbol{b_g} \in \mathbb{R}^{3\times1}$ are the velocity error-states expressed in the navigation frame, the misalignment error, accelerometer bias residual error, and gyroscope bias residual error, respectively. The linearized error state differential equation is

$$\boldsymbol{\delta\dot{x}^n} = \mathbf{F}\boldsymbol{\delta x^n} + \mathbf{G}\boldsymbol{n} \tag{47}$$

where $\boldsymbol{n} \in \mathbb{R}^{12\times1}$ is the system noise vector, $\mathbf{F} \in \mathbb{R}^{12\times12}$ is the system matrix, and $\mathbf{G} \in \mathbb{R}^{12\times12}$ is the system noise distribution matrix. There are several independent sources of noise in the system, each of which is assumed to have a zero mean Gaussian distribution as expressed in (38)-(39) and can be formulated into

$$\boldsymbol{n} = [\boldsymbol{n_a}^T \quad \boldsymbol{n_g}^T \quad \boldsymbol{n_{a_b}}^T \quad \boldsymbol{n_{g_b}}^T]^T \in \mathbb{R}^{12\times1}. \tag{48}$$

. The system matrix, $\mathbf{F}$, is given by:

$$\mathbf{F} = \begin{bmatrix} \mathbf{F}_{vv} & \mathbf{F}_{v\epsilon} & \mathbf{0}_{3\times3} & \mathbf{C}_b^n \\ \mathbf{F}_{\epsilon v} & \mathbf{F}_{\epsilon\epsilon} & \mathbf{C}_b^n & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{49}$$

and the exact sub-matrices can be found, for example, in [36]. In terms of the system noise distribution, matrix $\mathbf{G}$ is

$$\mathbf{G} = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{C}_b^n & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{C}_b^n & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \end{bmatrix} \tag{50}$$

where $\mathbf{0}_{3\times3}$ is a three-by-three zero matrix and $\mathbf{I}_{3\times3}$ is a three dimensional identity matrix. The measurement innovation, $\boldsymbol{\delta z}_k$, is the residual between the velocity as calculated by the INS velocity (31) and the velocity given by the DVL in the body frame:

$$\boldsymbol{\delta z} = \mathbf{C}_n^b \boldsymbol{v}_{INS}^n - \boldsymbol{v}_{DVL}^b. \tag{51}$$

By deriving the perturbation velocity error (51) and performing linearization, the measurement matrix, $\mathbf{H}_{DVL} \in \mathbb{R}^{3\times12}$, can be received:

$$\mathbf{H}_{DVL} = [\mathbf{C}_n^b \boldsymbol{v}_{INS}^n \quad -\mathbf{C}_n^b(\boldsymbol{v}_{INS}^n \times) \quad \mathbf{0}_{3\times3} \quad \mathbf{0}_{3\times3}]. \tag{52}$$

Once both the prediction and update steps are completed, corrections of the measurements occur in the following manner:

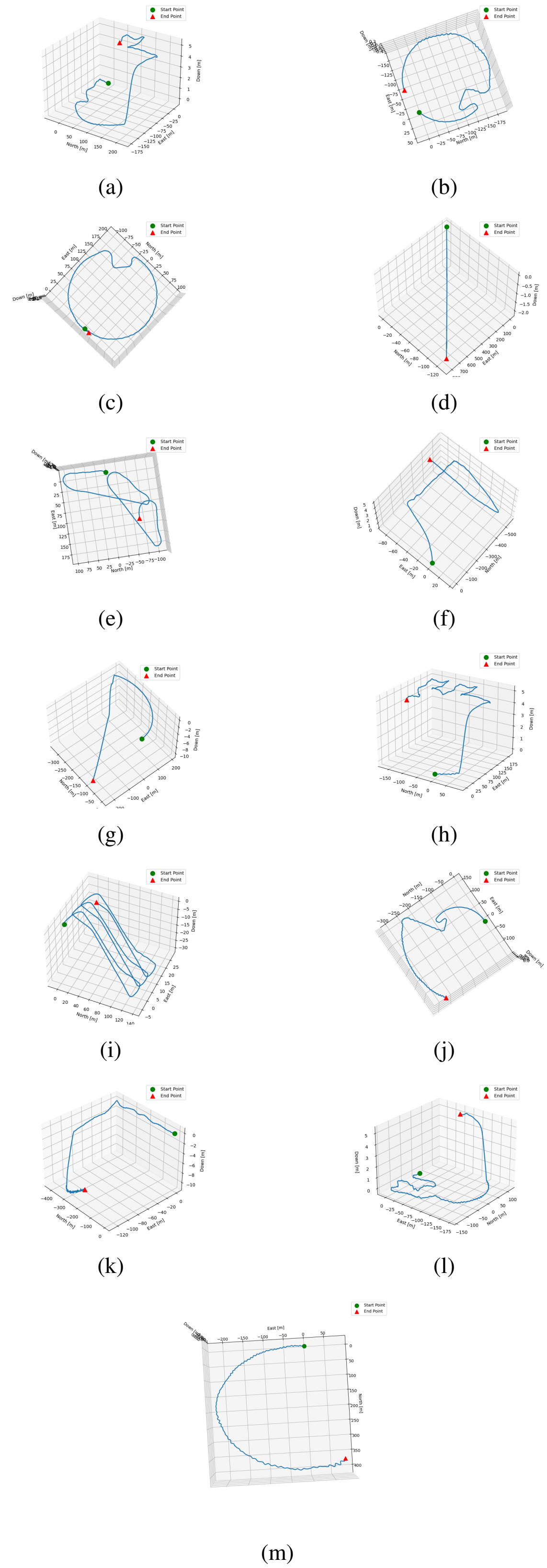
$$\boldsymbol{v}_{\boldsymbol{EKF}}^{\boldsymbol{n}} = \boldsymbol{v}_{\boldsymbol{INS}}^{\boldsymbol{n}} - \boldsymbol{\delta v^n} \tag{53a}$$
$$\hat{\mathbf{C}}_b^n = [\mathbf{I} - (\boldsymbol{\epsilon^n} \times)]\mathbf{C}_b^n \tag{53b}$$
$$\hat{\boldsymbol{b}}_{\boldsymbol{a}} = \boldsymbol{b_a} + \boldsymbol{\delta b_a} \tag{53c}$$
$$\hat{\boldsymbol{b}}_{\boldsymbol{g}} = \boldsymbol{b_g} + \boldsymbol{\delta b_g}. \tag{53d}$$

E. Implementation

As mentioned earlier, the proposed architecture, A-KIT, is a set-transformer tailored to process inertial and velocity time series data. The network takes a mini-batch comprising specific force, angular velocity from the IMU, and estimated velocity from the first version of the adaptive EKF. For each, a one-second window was taken, meaning one hundred samples. This data then undergoes the modified patch-embedding layer designed for time series data. Subsequently, it enters the set-transformer, as defined in Section IV. Additionally, the feed-forward network at the end has a slight modification: following the activation function, there is a dropout layer with a probability of 0.1, and the network's output passes through a ReLU activation function, ensuring that output values represent a positively defined matrix. However, any zero values the activation function produces are replaced with the value one. Lastly, the output of A-KIT is multiplied by the intermediate estimated innovation-based covariance matrix provided by (13) to generate the new discrete process noise covariance (29). An illustrative block diagram depicting our A-KIT approach in the INS/DVL case study is visualized in Fig 3. The hyperparameters used in the modified set-transformer A-KIT are presented in Table I. The optimizer employed is the RMSprop optimizer with a weight decay of 1×10^{-4}, a learning rate of 1×10^{-5}, and a mini-batch size of 400, trained over 100 epochs.

Hyper-parameter	α	β	γ	D	h	ffe	b	k
Value	5	1	1	32	2	64	2	1

TABLE I: The network's hyper-parameters and values.



(a) (b) (c) (d) (e) (f) (g) (h) (i) (j) (k) (l) (m)

Fig. 2: Thirteen AUV trajectories in the NED frame, measured in meters. Green circles denote starting points and red triangles signify endpoints. Trajectories (a)-(k) belong to the training set, while (l)-(m) comprise the test set.

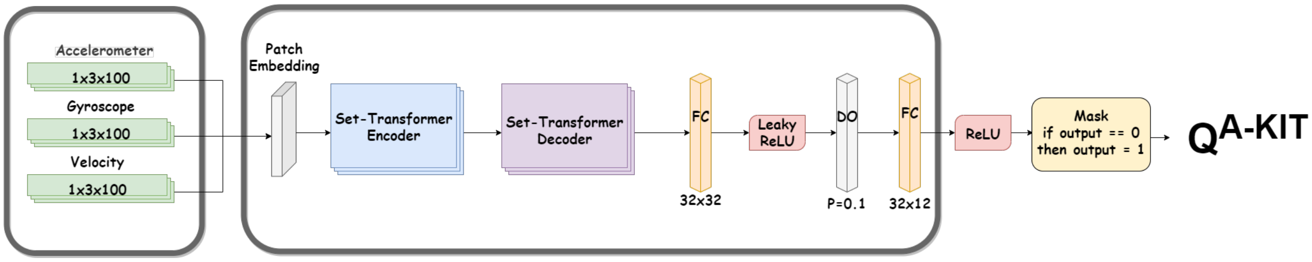


Fig. 3: A block diagram illustrating the set-transformer architecture for A-KIT INS/DVL fusion. The network takes a window of 100 samples from the inertial data along with velocity estimations from the EKF as input and produces the diagonal elements of the scale-factor matrix.

VII. Analysis and Results

A. Dataset Acquisition

We conducted experiments in the Mediterranean Sea near the shore of Haifa, Israel, using the Snapir AUV to gather data. Snapir is a modified, ECA Group, A18D mid-size AUV for deep-water applications. It performs autonomous missions up to 3000 meters in depth with 21 hours of endurance [54]. Snapir is equipped with iXblue Phins Subsea, which is a FOG-based high-performance subsea inertial navigation system [55] and Teledyne RDI Work Horse navigator DVL [56] that achieve accurate velocity measurements with a standard deviation of $0.02\ [m/s]$. The INS operates at a rate of 100 [Hz] and the DVL at 1 [Hz]. Fig. 4 displays the Snapir AUV during a mission.

The dataset was recorded on June 8^{th}, 2022, and contains approximately seven hours of data with different maneuvers, depths, and speeds. The train set is composed of eleven different data sections, each of a duration of $400\ [sec]$ and with different dynamics for diversity. We used only 73.3 minutes of the recorded data for training. As ground truth (GT), we used the filter solution given by Delph INS, post-processing software for iXblue's INS-based subsea navigation [57]. To evaluate the approach, we examined an additional two $400\ [sec]$ segments of the data that are not present in the training set, referring to them as the test set. The trajectories in the NED frame for each segment in both the training and test sets are illustrated in Fig. 2.

B. Competitive Approaches

The three variations of the model-based adaptive EKF (AEKF) were computed using an innovation window size of five updates. The initial $\delta\boldsymbol{x}_0$ and $\mathbf{P}_0^+$ for the conventional EKF are

$$\delta\boldsymbol{x}_0 = \begin{bmatrix}0.1[\frac{m}{sec}]\mathbf{1}_3 & 2.5^\circ\mathbf{1}_3 & 15[\mathrm{m}g]\mathbf{1}_3 & 15[\frac{\circ}{Hour}]\mathbf{1}_3\end{bmatrix} \quad (54)$$

$$\mathbf{P}_0^+ = \begin{bmatrix}0.2[\frac{m}{sec}]\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & 5^\circ\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & 30[\mathrm{m}g]\mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & 30[\frac{\circ}{Hour}]\mathbf{I}_3\end{bmatrix} \quad (55)$$

while for the adaptive EKF variations, these parameters are:

$$\delta\boldsymbol{x}_0 = \begin{bmatrix}0.1[\frac{m}{sec}]\mathbf{1}_3 & 0.5^\circ\mathbf{1}_3 & 15[\mathrm{m}g]\mathbf{1}_3 & 0.5[\frac{\circ}{Hour}]\mathbf{1}_3\end{bmatrix}. \quad (56)$$

Fig. 4: Snapir AUV being dispatched to a mission at sea.

$$\mathbf{P}_0^+ = \begin{bmatrix}0.2[\frac{m}{sec}]\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & 1^\circ\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & 30[\mathrm{m}g]\mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & 1[\frac{\circ}{Hour}]\mathbf{I}_3\end{bmatrix} \quad (57)$$

These values were determined through manual optimization to yield the best results with respect to the VRMSE metric. The first AEKF, denoted as AEKF1, is defined in (13), AEKF2 in (16), and AEKF3 in (17). For the latter, a forgetting factor of γ = 0.15 was chosen.

C. Evaluation Metrics

To assess the performance of the proposed approach, we employed the root mean square error (RMSE) metric, examining both velocity RMSE (VRMSE) and position RMSE (PRMSE), defined as follows:

$$PRMSE(\boldsymbol{p}_i, \hat{\boldsymbol{p}}_i) = \sqrt{\frac{\sum_{i=1}^{N}(\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i)^2}{N}} \quad (58)$$

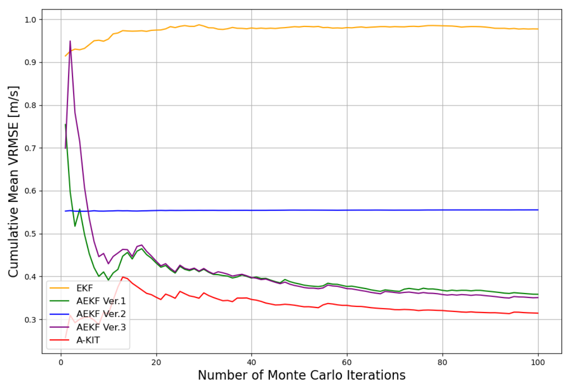
where N is the number of samples, $\boldsymbol{p}_i$ denotes the ground truth position vector, and $\hat{\boldsymbol{p}}_i$ represents the predicted position vector.

$$VRMSE(\boldsymbol{v}_i, \hat{\boldsymbol{v}}_i) = \sqrt{\frac{\sum_{i=1}^{N}(\boldsymbol{v}_i - \hat{\boldsymbol{v}}_i)^2}{N}} \quad (59)$$

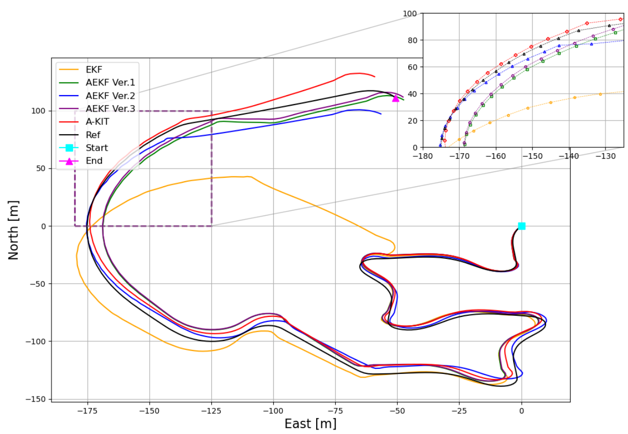
where $\boldsymbol{v}_i$ denotes the ground truth velocity vector, and $\hat{\boldsymbol{v}}_i$ represents the predicted velocity vector.

D. Experimental Results

The proposed A-KIT approach underwent evaluation using test set trajectories depicted in Fig. 2, comparing its performance against the conventional EKF and three AEKFs. A Monte Carlo (MC) test consisting of 100 runs was made, with initial conditions drawn from a Gaussian distribution and $\mathcal{N}(\delta\boldsymbol{x}_0, \mathbf{P}_0)$ as defined in (54)-(57).



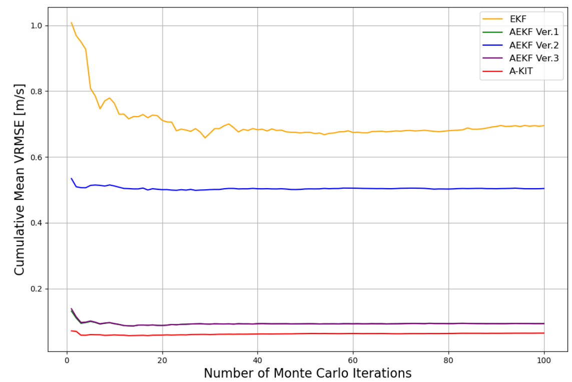
(a)



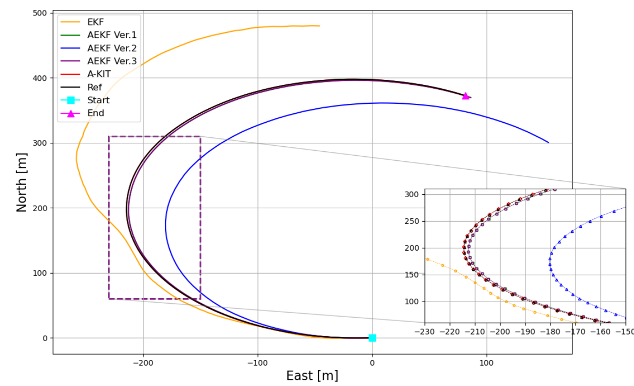
(b)

Fig. 5: (a) The cumulative mean of the VRMSE as a function of the Monte Carlo iteration for the (l) trajectory. In (b), a trajectory comparison is presented between the EKF variations for the (l) trajectory.

Figs. 5(b) and 6(b) display the position comparison for each method for test trajectories (l) and (m). In Fig.5(a), the cumulative mean of the VRMSE for trajectory (l) is presented as a function of the MC iterations. Convergence to a steady state is observed for all approaches, with the A-KIT outperforming both conventional and adaptive EKF versions. The A-KIT demonstrates a 67.8% improvement in VRMSE and a 49.5% improvement in PRMSE compared to the EKF. Additionally, it exhibits an improvement of more than 10% and 8% for the VRMSE and PRMSE of the adaptive approaches, respectively. In the zoomed-in portion of 5(b), the suggested approach is shown to closely follow the trajectory compared to the others.



(a)



(b)

Fig. 6: (a) The cumulative mean of the VRMSE as a function of the Monte Carlo iteration for the (m) trajectory. In (b), a trajectory comparison is presented between the EKF variations for the (m) trajectory.

When examining trajectory (m), Fig. 6(a) demonstrates similar behavior as for the (l) trajectory. All the filters converge to a steady state, and the A-KIT possesses the lower cumulative mean VRMSE. In this case, the results are even more superior than the previous ones. An improvement of more than 90% of the A-KIT over the EKF with respect to the VRMSE and 87.7% with respect to the PRMSE is observed. Furthermore, there is an improvement of more than 31% for the A-KIT when compared to the VRMSE of the adaptive approaches and more than 37% when compared to the PRMSE. All the results are organized in Table II, where the exact values of the VRMSE and PRMSE are provided.

VIII. Conclusions

This paper addresses the crucial task of applying an online adaptive navigation filter. To that end, we proposed A-KIT, a hybrid learning framework. Our approach leverages the advantages of DL, specifically a set-transformer network, to

Method	VRMSE [m/sec]		A-KIT VRMSE Improvement		PRMSE [m]		A-KIT PRMSE Improvement	
Trajectory	(l)	(m)	(l)	(m)	(l)	(m)	(l)	(m)
EKF	0.977	0.694	67.8 %	90.7 %	35.5	80.05	49.5 %	87.7 %
AEKF ver. 1	0.358	0.0934	12.29 %	31.47 %	19.6	15.567	8.6 %	37.11 %
AEKF ver. 2	0.555	0.503	43.4 %	87.2 %	26.1	66.4	31.4 %	85.2 %
AEKF ver. 3	0.350	0.0931	10.28 %	31.2 %	20.7	15.54	13.5 %	37 %
A-KIT (Ours)	0.314	0.064	N/A	N/A	17.9	9.79	N/A	N/A

TABLE II: PRMSE and VRMSE performance of the conventional EKF (fixed process noise covariance), three AEKFs, and our proposed approach.

dynamically adapt the process noise covariance matrix of an EKF. We rely on the well-established EKF theory and enjoy the benefits of merging deep-learning algorithms with the filter. The proposed Kalman-informed loss enables seamless integration between the DL approach and the model-based EKF. Utilizing this loss, the network can be optimized to provide a covariance noise matrix that minimizes errors in the desired state while adhering to all Kalman methodologies.
A-KIT was developed as a versatile algorithm applicable to various navigation-related fusion tasks. To demonstrate its effectiveness, we conducted a case study focused on the nonlinear fusion of INS/DVL in an AUV. A-KIT was compared against four approaches, including the conventional EKF with a constant process noise covariance along the trajectory and three different adaptive EKFs that dynamically adjust the process noise covariance based on the filter's innovation. A-KIT was trained and evaluated using real recorded data from an AUV spanning 86.6 minutes. The results illustrate that A-KIT outperformed all other adaptive approaches, showcasing an average improvement of 35.4% in PRMSE and 35.9% in VRMSE.

Acknowledgment

N.C. is supported by the Maurice Hatter Foundation and University of Haifa presidential scholarship for outstanding students on a direct Ph.D. track.

References

[1] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: Theory algorithms and software*. John Wiley & Sons, 2001.

[2] M. I. Ribeiro, "Kalman and extended Kalman filters: Concept, derivation and properties," *Institute for Systems and Robotics*, vol. 43, no. 46, pp. 3736–3741, 2004.

[3] E. A. Wan and R. Van Der Merwe, "The unscented Kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*. IEEE, 2000, pp. 153–158.

[4] F. Daum, "Nonlinear filters: Beyond the Kalman filter," *IEEE Aerospace and Electronic Systems Magazine*, vol. 20, no. 8, pp. 57–69, 2005.

[5] H. Heffes, "The effect of erroneous models on the Kalman filter response," *IEEE Transactions on Automatic Control*, vol. 11, no. 3, pp. 541–543, 1966.

[6] S. Sangsuk-Iam and T. E. Bullock, "Analysis of discrete-time Kalman filtering under incorrect noise covariances," *IEEE Transactions on Automatic Control*, vol. 35, no. 12, pp. 1304–1309, 1990.

[7] S. S. Saab, "Discrete-time Kalman filter under incorrect noise covariances," in *Proceedings of 1995 American Control Conference-ACC'95*, vol. 2. IEEE, 1995, pp. 1152–1156.

[8] D. Simon, *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[9] J. Farrell, *Aided navigation: GPS with high rate sensors*. McGraw-Hill, Inc., 2008.

[10] R. Mehra, "On the identification of variances and adaptive Kalman filtering," *IEEE Transactions on Automatic Control*, vol. 15, no. 2, pp. 175–184, 1970.

[11] L. Zhang, D. Sidoti, A. Bienkowski, K. R. Pattipati, Y. Bar-Shalom, and D. L. Kleinman, "On the identification of noise covariances and adaptive Kalman filtering: A new look at a 50 year-old problem," *IEEE Access*, vol. 8, pp. 59 362–59 388, 2020.

[12] Y. Li, R. Chen, X. Niu, Y. Zhuang, Z. Gao, X. Hu, and N. El-Sheimy, "Inertial sensing meets machine learning: Opportunity or challenge?" *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[13] N. Cohen and I. Klein, "Inertial Navigation Meets Deep Learning: A Survey of Current Trends and Future Directions," *arXiv preprint arXiv:2307.00014*, 2023.

[14] A. B. Chatfield, *Fundamentals of high accuracy inertial navigation*. AIAA, 1997, vol. 174.

[15] D. Titterton, J. L. Weston, and J. Weston, *Strapdown inertial navigation technology*. IET, 2004, vol. 17.

[16] A. Abosekeen, A. Noureldin, and M. J. Korenberg, "Improving the RISS/GNSS land-vehicles integrated navigation system using magnetic azimuth updates," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1250–1263, 2019.

[17] A. Mohamed and K. Schwarz, "Adaptive Kalman filtering for INS/GPS," *Journal of Geodesy*, vol. 73, pp. 193–203, 1999.

[18] C. Hide, T. Moore, and M. Smith, "Adaptive Kalman filtering for low-cost INS/GPS," *The Journal of Navigation*, vol. 56, no. 1, pp. 143–152, 2003.

[19] 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.

[20] H. Shen, G. Wen, Y. Lv, J. Zhou, and L. Wang, "USV parameter estimation: Adaptive unscented Kalman filter-based approach," *IEEE Transactions on Industrial Informatics*, 2022.

[21] W. Gao, J. Li, G. Zhou, and Q. Li, "Adaptive Kalman filtering with recursive noise estimator for integrated SINS/DVL systems," *The Journal of Navigation*, vol. 68, no. 1, pp. 142–161, 2015.

[22] D. Wang, X. Xu, and L. Hou, "An improved adaptive Kalman filter for underwater SINS/DVL system," *Mathematical Problems in Engineering*, vol. 2020, pp. 1–14, 2020.

[23] D. C. Silva, A. Frutuoso, L. F. Souza, and E. A. de Barros, "Comparative Analysis of Innovation-Based Adaptive Kalman Filters Applied to AUVs Navigation," in *2022 Latin American Robotics Symposium (LARS), 2022 Brazilian Symposium on Robotics (SBR), and 2022 Workshop on Robotics in Education (WRE)*. IEEE, 2022, pp. 31–36.

[24] Q. Wang, K. Liu, and Z. Cao, "System noise variance matrix adaptive Kalman filter method for AUV INS/DVL navigation system," *Ocean Engineering*, vol. 267, p. 113269, 2023.

[25] B. Or and I. Klein, "ProNet: Adaptive process noise estimation for INS/DVL fusion," in *2023 IEEE Underwater Technology (UT)*, 2023, pp. 1–5.

[26] M. Brossard, A. Barrau, and S. Bonnabel, "AI-IMU dead-reckoning," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 4, pp. 585–595, 2020.

[27] X. Gao, H. Luo, B. Ning, F. Zhao, L. Bao, Y. Gong, Y. Xiao, and J. Jiang, "RL-AKF: An adaptive Kalman filter navigation algorithm based on reinforcement learning for ground vehicles," *Remote Sensing*, vol. 12, no. 11, p. 1704, 2020.

[28] F. Wu, H. Luo, H. Jia, F. Zhao, Y. Xiao, and X. Gao, "Predicting the noise covariance with a multitask learning model for Kalman filter-based GNSS/INS integrated navigation," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–13, 2020.

[29] Y. Xiao, H. Luo, F. Zhao, F. Wu, X. Gao, Q. Wang, and L. Cui, "Residual attention network-based confidence estimation algorithm for non-holonomic constraint in GNSS/INS integrated navigation system," *IEEE Transactions on Vehicular Technology*, vol. 70, no. 11, pp. 11 404–11 418, 2021.

[30] Z. Zou, T. Huang, L. Ye, and K. Song, "CNN based adaptive Kalman filter in high-dynamic condition for low-cost navigation system on highspeed UAV," in *2020 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. IEEE, 2020, pp. 103–108.

[31] B. Or and I. Klein, "A hybrid model and learning-based adaptive navigation filter," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–11, 2022.

[32] R. Huang, M. Lei, X. Zhang, L. Zhou, Y. Lu, and B. He, "LSTM-based Process Noise Covariance Prediction for AUV Navigation," in *2023 IEEE 7th Information Technology and Mechatronics Engineering Conference (ITOEC)*, vol. 7. IEEE, 2023, pp. 1657–1661.

[33] B. Or and I. Klein, "Adaptive step size learning with applications to velocity aided inertial navigation system," *IEEE Access*, vol. 10, pp. 85 818–85 830, 2022.

[34] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop KF: Learning discriminative deterministic state estimators," *Advances in Neural Information Processing Systems*, vol. 29, 2016.

[35] S. Jouaber, S. Bonnabel, S. Velasco-Forero, and M. Pilte, "Nnakf: A neural network adapted Kalman filter for target tracking," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 4075–4079.

[36] P. D. Groves, "Principles of GNSS, inertial, and multisensor integrated navigation systems," *IEEE Aerospace and Electronic Systems Magazine*, vol. 30, no. 2, pp. 26–27, 2015.

[37] A. Gelb *et al.*, *Applied optimal estimation*. MIT Press, 1974.

[38] R. Mehra, "Approaches to adaptive filtering," *IEEE Transactions on Automatic Control*, vol. 17, no. 5, pp. 693–698, 1972.

[39] W. Ding, J. Wang, C. Rizos, and D. Kinlyside, "Improving adaptive Kalman estimation in GPS/INS integration," *The Journal of Navigation*, vol. 60, no. 3, pp. 517–529, 2007.

[40] A. Almagbile, J. Wang, and W. Ding, "Evaluating the performances of adaptive Kalman filter methods in GPS/INS integration," *Journal of Global Positioning Systems*, vol. 9, no. 1, pp. 33–40, 2010.

[41] S. Akhlaghi, N. Zhou, and Z. Huang, "Adaptive adjustment of noise covariance in Kalman filter for dynamic state estimation," in *2017 IEEE power & energy society general meeting*. IEEE, 2017, pp. 1–5.

[42] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly *et al.*, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[43] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.

[44] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[45] J. Lee, Y. Lee, J. Kim, A. Kosiorek, S. Choi, and Y. W. Teh, "Set transformer: A framework for attention-based permutation-invariant neural networks," in *International conference on machine learning*. PMLR, 2019, pp. 3744–3753.

[46] R. M. Rogers, *Applied mathematics in integrated navigation systems*. AIAA, 2003, vol. 1.

[47] E.-H. Shin and N. El-Sheimy, "Accuracy improvement of low cost INS/GPS for land applications," in *Proceedings of the 2002 National Technical Meeting of the Institute of Navigation*, 2002, pp. 146–157.

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

[49] D. Rudolph and T. A. Wilson, "Doppler Velocity Log theory and preliminary considerations for design and construction," in *2012 Proceedings of IEEE Southeastcon*. IEEE, 2012, pp. 1–7.

[50] A. Tal, I. Klein, and R. Katz, "Inertial navigation system/Doppler velocity log (INS/DVL) fusion with partial DVL measurements," *Sensors*, vol. 17, no. 2, p. 415, 2017.

[51] 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.

[52] F. L. Markley, "Attitude determination using vector observations and the singular value decomposition," *Journal of the Astronautical Sciences*, vol. 36, no. 3, pp. 245–258, 1988.

[53] I. Klein and R. Diamant, "Observability analysis of DVL/PS aided INS for a maneuvering AUV," *Sensors*, vol. 15, no. 10, pp. 26 818–26 837, 2015.

[54] ECA Group, "A18-D AUV: Autonomous Underwater Vehicle," https://www.ecagroup.com/en/solutions/a18-d-auv-autonomous-underwater-vehicle, Accessed: September 2023.

[55] iXblue, "PHINS Subsea," https://www.ixblue.com/store/phins-subsea/, Accessed: September 2023.

[56] Teledyne Marine, "Doppler Velocity Logs," https://www.teledynemarine.com/products/product-line/navigation-positioning/doppler-velocity-logs, Accessed: September 2023.

[57] iXblue, "DELPH INS," https://www.ixblue.com/store/delph-ins/, Accessed: September 2023.