

Supplementary Information for “Kinematic-Aware AIS Preprocessing and Bidirectional Recurrent Modeling for Multi-Output Vessel Trajectory Prediction”

Marilena Sinni Dimitris M. Kyriazanos

Institute of Informatics & Telecommunications,
National Centre for Scientific Research “Demokritos”, Athens, Greece

This document specifies every parameter, hyperparameter, and implementation detail required to reproduce the results of the main article. Section and equation numbers of the form (11), 3.1, etc. refer to the main article; numbers prefixed with S refer to this document.

S1 Coordinate reference systems

Raw AIS positions are reported in WGS84 geographic coordinates (EPSG:4326). All metric computations in the preprocessing pipeline, the displacement test of Eqs. (3)–(4), the correction of Eqs. (5)–(6), and the linear interpolation applied during 1-minute resampling, are carried out in projected UTM coordinates, in which distances are Euclidean in metres. A single UTM zone is used per dataset: zone 30N (EPSG:32630) for Brest and zone 32N (EPSG:32632) for the Kattegat region of interest. Corrected and resampled positions are transformed back to WGS84 before the next stage. The model inputs (Section 3.2 of the main article) are z-score-normalised longitude and latitude; no projection is applied before the network, and the conversion to metres occurs only in the loss and the evaluation metrics, through the local equirectangular projection given below.

The eastern edge of the DMA bounding box (13°E) extends 1° beyond the nominal limit of zone 32N (12°E). The resulting scale distortion is below 0.04%, i.e. under 0.1 m on a typical one-minute displacement, and therefore immaterial at the error levels reported in the main article.

Coordinate transformations between geographic and planar coordinates are performed with `pyproj` 3.7.2, using a UTM projection on the WGS-84 ellipsoid (zone 30N for Brest, zone 32N for DMA). These are the coordinates in which the maximum-admissible-displacement test, the linear position-correction interpolation, and the uniform one-minute resampling are carried out; all three are stated in full in the main paper.

The complementary evaluation metrics and the position term of the composite training loss, also given in the main paper, instead use a local equirectangular projection centred on the true position of each sample, which removes any dependence on a fixed UTM zone. With $\Delta\lambda$ and $\Delta\varphi$ the longitude and latitude differences in degrees between the predicted and the true position, and φ_0 the true latitude, the planar offsets in meters are

$$x = k \Delta\lambda \cos \varphi_0, \quad y = k \Delta\varphi, \quad (\text{S1})$$

where $k = 111\,320$ m per degree of arc, corresponding to an effective Earth radius $R = k \cdot 180/\pi \approx 6.378 \times 10^6$ m, the WGS-84 semi-major axis. Since φ_0 is re-evaluated for each sample, the longitudinal scale factor always matches the local latitude, keeping the approximation accurate over the displacement magnitudes involved.

S2 Preprocessing parameters

Table S1 lists every parameter of the preprocessing pipeline of Section 3.1 of the main article. The pipeline is applied identically to both corpora; only the geographic bounding box differs and is supplied through the configuration layer.

Table S1: Complete preprocessing parameter set. “–” = stage disabled / not applicable for that dataset.

Parameter	Brest	DMA
<i>Data cleaning</i>		
Geographic bounding box (lat, lon)	–	[55.5°, 58°], [10.3°, 13°]
Duplicate criterion	exact duplicates; same-MMSI records < 1 s apart	
Valid COG range	[0°, 360°]; missing COG removed	
Retained SOG range	(0, 30] knots (SOG = 0 and SOG > 30 removed)	
Valid ship-type codes	[20, 90]; vessels outside range excluded entirely	
<i>Trajectory extraction</i>		
Segmentation gap threshold	10 minutes	
Maximum trajectory duration	24 hours (longer tracks split into 24-h windows)	
Maximum points per segment	500 (longer segments subdivided approx. equally)	
Minimum points per segment	25 (shorter segments discarded)	
Stationary filter	mean SOG < 1 knot ⇒ trajectory removed	
<i>Speed anomaly correction (Eqs. 1–2)</i>		
$a_{\max} = d_{\max}$	0.05 knots/s	
Correction	mean of the two neighbouring SOG values	
Passes	1	
<i>Position anomaly correction (Eqs. 3–6, Algorithm S2)</i>		
$a_{\max}$	$0.05 \times 0.5144 \text{ m/s}^2$	
Knots-to-m/s factor	0.5144	
Δt^{cap}	$\min(\Delta t, 60) \text{ s}$	
Flagging rule	both incoming and outgoing segments infeasible	
Correction	time-linear interpolation between neighbours (UTM)	
Safeguard	revert if corrected lat/lon leaves valid domain	
Maximum passes	3, with early termination on a pass with no corrections	
<i>Resampling</i>		
Target interval	60 s (uniform)	
Position interpolation	linear, in UTM	
SOG	recomputed from interpolated positions as $\Delta d / \Delta t$	
COG	unwrap → linear interpolation → rewrap to [0°, 360°)	
Post-filter	recomputed SOG outside (0, 30] kn or invalid coords discarded	

S3 Anomaly correction pseudocode

Algorithms S1 and S2 give the exact procedures for the speed and iterative position anomaly corrections of Section 3.1. Both operate per trajectory on chronologically sorted points. Endpoint samples ($i = 1$ and $i = n$) are never modified, since the correction requires both neighbours.

Algorithm S1 Speed anomaly correction (single pass, per trajectory)

Require: points (t_i, SOG_i) , $i = 1, \dots, n$; $a_{\max} = d_{\max} = 0.05 \text{ kn/s}$

```
1: for  $i = 2$  to  $n - 1$  do
2:    $\Delta t \leftarrow t_i - t_{i-1}$ 
3:   if  $|\text{SOG}_i - \text{SOG}_{i-1}| > a_{\max} \Delta t$  then
4:      $\text{SOG}_i \leftarrow (\text{SOG}_{i-1} + \text{SOG}_{i+1})/2$ 
5:   end if
6: end for
```

Algorithm S2 Iterative position anomaly correction (per trajectory)

Require: points $(t_i, x_i, y_i, \text{SOG}_i)$ in UTM, $i = 1, \dots, n$; $a_{\max} = 0.05 \times 0.5144 \text{ m/s}^2$; $c = 0.5144$; $\text{cap} = 60 \text{ s}$; $K = 3$

```
1: for pass = 1 to  $K$  do
2:   corrections  $\leftarrow 0$ 
3:   for  $i = 2$  to  $n - 1$  do
4:      $\Delta t_{\text{in}} \leftarrow t_i - t_{i-1}$ ;  $\Delta t_{\text{out}} \leftarrow t_{i+1} - t_i$ 
5:      $d_{\text{in}}^{\max} \leftarrow c \text{SOG}_{i-1} \Delta t_{\text{in}} + \frac{1}{2} a_{\max} \min(\Delta t_{\text{in}}, \text{cap})^2$ 
6:      $d_{\text{out}}^{\max} \leftarrow c \text{SOG}_i \Delta t_{\text{out}} + \frac{1}{2} a_{\max} \min(\Delta t_{\text{out}}, \text{cap})^2$ 
7:      $d_{\text{in}} \leftarrow \|(x_i, y_i) - (x_{i-1}, y_{i-1})\|_2$ ;  $d_{\text{out}} \leftarrow \|(x_{i+1}, y_{i+1}) - (x_i, y_i)\|_2$ 
8:     if  $d_{\text{in}} > d_{\text{in}}^{\max}$  and  $d_{\text{out}} > d_{\text{out}}^{\max}$  then
9:        $\alpha \leftarrow (t_i - t_{i-1}) / (t_{i+1} - t_{i-1})$ 
10:       $(x', y') \leftarrow (x_{i-1}, y_{i-1}) + \alpha [(x_{i+1}, y_{i+1}) - (x_{i-1}, y_{i-1})]$ 
11:       $(\varphi', \lambda') \leftarrow \text{UTM} \rightarrow \text{WGS84}(x', y')$ 
12:      if  $\varphi' \in [-90^\circ, 90^\circ]$  and  $\lambda' \in [-180^\circ, 180^\circ]$  then
13:         $(x_i, y_i) \leftarrow (x', y')$ ; corrections  $\leftarrow$  corrections + 1
14:      end if
15:    end if
16:  end for
17:  if corrections = 0 then break
18: end if
19: end for
```

S4 Model architectures

All nine models share the input representation of Section 3.2 ($T = 5$, stride 1, feature vector $p'_t = (x_t, y_t, v_t, \cos \psi_t, \sin \psi_t)$ with x, y, v z-score normalised on training-split statistics), the three-head decoder, the loss (Eq. 11), and the training configuration of Section S6. Table S2 consolidates the architectural hyperparameters; parameter counts are given in Table 1 of the main article.

Attention block (Att- k variants). A single scaled dot-product multi-head self-attention sublayer operates on the sequence of per-step encoder outputs $e_t \in \mathbb{R}^d$ ($t = 1, \dots, T$), with queries, keys, and values obtained by learned linear projections (with bias) of the full width d , split evenly across $k \in \{2, 4\}$ heads ($d_{\text{head}} = d/k$), followed by an output projection $d \rightarrow d$. Attention is unmasked, so every step attends to the whole window in both temporal directions. The sublayer output is combined with its input in post-normalisation form,

$$\tilde{e}_t = \text{LN}(e_t + \text{MHSA}(e_{1:T})_t), \quad (\text{S2})$$

and the state at the final time step, $\tilde{e}_T$, forms the decoder context, replacing Eq. (10). The block contains no position-wise feed-forward sublayer. No positional encoding is added, since

Table S2: Architectural hyperparameters common to the model comparison.

Component	Setting
Recurrent encoder	1 layer; hidden size $h = 256$ per direction
Encoder output width	$d = 2h = 512$ (bidirectional); $d = h = 256$ (unidirectional)
Context vector	concat of final forward and final backward states (Eq. 10)
Position head	Linear($d \rightarrow d/2$) – ReLU – Linear($d/2 \rightarrow 2$)
Course head	Linear($d \rightarrow d/2$) – ReLU – Linear($d/2 \rightarrow 2$)
Speed head	Linear($d \rightarrow 1$)
Dropout	none
Weight initialisation	PyTorch defaults
Bias initialisation	PyTorch defaults

the attention operates on recurrent states that are already order-dependent. No dropout is applied within the block.

S5 Loss function

The training objective is Eq. (11) of the main article, restated here with every constant:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \left[w_p \frac{(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2}{D^2} + w_v (v_i - \hat{v}_i)^2 + \frac{w_\psi}{2} (1 - \cos \Delta\psi_i) \right], \quad (\text{S3})$$

with $(w_p, w_v, w_\psi) = (0.7, 0.2, 0.1)$, fixed a priori and not tuned, and reference distance $D = 500$ m. The position residuals are expressed in metres in the local equirectangular projection of Section S1 after de-normalising the network outputs; v is the z-score-normalised SOG; $\cos \Delta\psi_i$ is the cosine similarity between the predicted course vector $(\widehat{\cos \psi_i}, \widehat{\sin \psi_i})$ and the true unit vector. The same objective, with the same weights, is used unchanged for every architecture and every ablation variant.

Evaluated on the held-out Brest test split with the five cross-validation checkpoints of the BiGRU model ($N = 320,710$ windows per fold), the three weighted terms of Eq. (11) are 0.0057 ± 0.0006 (position), 0.0103 ± 0.0003 (speed), and 0.0034 ± 0.0000 (course), summing to 0.0194 ± 0.0008 ; values are mean $\pm$ standard deviation across folds. The speed term therefore accounts for roughly half of the converged objective despite carrying the smallest of the three residual magnitudes before weighting, while the position term contributes under a third even at $w_p = 0.7$.

S6 Training configuration

Table S3: Complete training configuration, identical for every architecture and ablation variant.

Setting	Value
Optimiser	Adam
Initial learning rate	10^{-4}
$\beta_1, \beta_2, \varepsilon$	0.9, 0.999, 10^{-8} (PyTorch defaults)
Weight decay	0
LR schedule	cosine annealing, $T_{\max} = 200$ epochs, $\eta_{\min} = 10^{-6}$
Batch size	128
Maximum epochs	200
Early stopping	20 epochs without improvement of validation loss (Eq. 11)
Model selection	weights of the best validation epoch
Gradient clipping	none
Numerical precision	float32 (no mixed precision)
Shuffling	training windows shuffled each epoch; validation/test order fixed
Random seed(s)	42 for all data partitioning (train/val/test split and k -fold assignment), shared across architectures and folds; weight initialisation and batch ordering are not seeded

S7 Data partitioning and cross-validation

Each dataset is split at the trajectory level into 70% training, 10% validation, and 20% test partitions. Whole trajectories are assigned at random under fixed seed 42, so that no trajectory contributes points to more than one partition. The split is not temporal: all three partitions span the full observation period. The identical partition, generated from the same seed, is used for every architecture and every ablation variant.

The five cross-validation folds are drawn from the combined training and validation pool (80% of trajectories): the pool is partitioned at random (same seed, 42) into five disjoint trajectory subsets of equal size, each serving in turn as the validation split while the remaining four form the training split. Each fold therefore trains on 64% and validates on 16% of all trajectories. Normalisation statistics are re-estimated on the training trajectories of each fold. The held-out test partition is identical across folds, so the reported inter-fold dispersion reflects the training/validation composition and optimisation stochasticity alone.

Table S4: Partition sizes after preprocessing. Window counts refer to length-5 sliding windows with stride 1.

	Brest	DMA
Trajectories, total	25,660	114,997
Trajectories, train (70%)	17,962	80,497
Trajectories, validation (10%)	2,566	11,500
Trajectories, test (20%)	5,132	23,000
Windows (samples), train	1,126,086	4,671,239
Windows (samples), validation	162,529	670,727
Windows (samples), test	320,710	1,337,399

S8 Evaluation metrics

All metrics are computed per predicted sample on the test partition, in physical units.

- **Position error:** Euclidean distance in metres between predicted and true coordinates, under the same local equirectangular projection used in the position term of the loss.
- **Directional error:** angle in degrees between the predicted and true displacement vectors taken from the last observed position. Samples with negligible movement, for which the angle is undefined, are excluded: a sample is discarded whenever the norm of either the true or the predicted displacement falls below 10^{-6} degrees of coordinate difference, i.e., below approximately 0.1 m at the latitudes of both study areas.
- **SOG error:** absolute deviation in knots.
- **COG error:** minimal angular deviation in degrees, wrapped to $[0^\circ, 180^\circ]$.

Aggregation is two-stage: within each fold, per-sample errors are summarised by their median; the five per-fold medians are then summarised across folds by their mean and standard deviation. Every tabulated value in the main article is the across-fold mean $\pm$ std of a per-fold median.

S9 Software and hardware environment

Table S5: Software and hardware used for all experiments.

Component	Version / specification
GPUs	2× NVIDIA GeForce RTX 2080 (8 GB GDDR6 each); each run used a single GPU
GPU driver	NVIDIA 550.163.01
CPU	AMD Ryzen Threadripper 2950X (16 cores / 32 threads)
RAM	64 GB
Operating system	Debian GNU/Linux 13 (trixie), kernel 6.12.57
Python	3.13.11
PyTorch	2.10.0 (+cu128)
CUDA / cuDNN	12.8 / 9.10.2
NumPy	2.4.3
pandas	3.0.1
scikit-learn	1.8.0
SciPy	1.17.1
pyproj / PROJ	3.7.2 / 9.5.1
Training time per epoch (BiGRU, Brest)	≈ 1.1 min (≈ 65 s)
Total training time (BiGRU, one fold, Brest)	≈ 3.3 h (3.6 h for a full 200-epoch fold)
Total training time (BiGRU, 5 folds, Brest)	≈ 16.5 h