2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

Neural-Kalman GNSS/INS Navigation for Precision Agriculture

Yayun Du[1*], Swapnil Sayan Saha[2*], Sandeep Singh Sandha[3], Arthur Lovekin[1], Jason Wu[2], S. Siddharth[4], Mahesh Chowdhary[4], Mohammad Khalid Jawed[1], Mani Srivastava[2]

***Abstract*— Precision agricultural robots require high-resolution navigation solutions. In this paper, we introduce a robust neural-inertial sequence learning approach to track such robots with ultra-intermittent GNSS updates. First, we propose an ultra-lightweight neural-Kalman filter that can track agricultural robots within 1.4 m (1.4 - 5.8$\times$ better than competing techniques), while tracking within 2.75 m with 20 mins of GPS outage. Second, we introduce a user-friendly video-processing toolbox to generate high-resolution ($\pm$5 cm) position data for fine-tuning pre-trained neural-inertial models in the field. Third, we introduce the first and largest (6.5 hours, 4.5 km, 3 phases) public neural-inertial navigation dataset for precision agricultural robots. The dataset, toolbox, and code are available at: https://github.com/nesl/agrobot.**

I. Introduction

Midway through the 1980s, precision agriculture (PA) emerged as a technique for timely, granular, and data-driven management of farms, livestock, orchards, and turfs [1]. PA optimizes production, minimizes waste, and reduces financial costs while enhancing environmental sustainability and food security [2]. Mobile robots, equipped with various sensors (e.g., soil moisture), actuators (e.g., delicate fruit-picking end-effectors), and machine-learning algorithms (e.g., weed classification), form key aids in automatically handling variability and uncertainties in agricultural practices [3][4][5][6].

Identifying precise points of interest in the field requires a high-resolution navigation solution for agricultural robots [7]. Large-scale agricultural robots use bulky and power-hungry RTK GPS/GNSS systems to navigate with a resolution of $\pm$0.4 m [7][8]. This approach is not suitable for agricultural robots designed to operate in row crops (e.g., flaxseed and canola) where the crop line spacing is less than 0.3 meters, or the robot must travel under crop canopy (GPS-denied) [9]. Such robots have a limited size, computing, and power budget, yet require a high precision navigation solution not achievable by GPS alone [9]. Proposed alternatives include using wheel encoders [10] or cameras (visual odometry) [11]. The former suffers from wheel slippage in muddy patches [12], while the latter suffers from changes in ambient lighting, occlusions, excessive power consumption, and terrain bumps [9]. Using quadrotors and radio beacons has coverage and infrastructure setup issues, which may not scale well across different fields [13]. Another alternative fuses inertial navigation systems (INS) with GPS/GNSS updates using Kalman filters [14][15][16]. While inertial sensors have a small footprint, low delay, and low power [11][17], existing GNSS/INS frameworks for PA use heuristic propagation models that are over-reliant on GPS updates to prevent position drift.

Here, we introduce robust **neural-inertial (NI) sequence learning** [17][18] as the propagation model in GNSS/INS Kalman fusion for PA. Instead of using human-engineered models, a neural network predicts the velocity and location of the robot during Kalman propagation from raw inertial measurement unit (IMU) readings. The robot can navigate longer without GPS updates over heuristic-based techniques [17][19][20][21][22]. Nonetheless, using pre-trained NI localization models requires 1-20 minutes of labeled high-resolution ground truth position data in the target field [17], which is hard to obtain by physically visiting the field. We introduce a video-processing pipeline that allows farmers to automatically extract position from raw quadrotor video feeds and use the labeled data to fine-tune pre-trained models. We also release the largest public NI navigation dataset for mobile robots for PA. Our contributions are as follows:

- *PANI* dataset: A dataset containing 6 hours (4.5 km) of high-fidelity IMU, GPS, and ground truth position data from a real PA mobile robot for training NI models;
- An automated video-processing pipeline to extract ground truth position data from quadrotor videos to fine-tune pre-trained odometry models;
- A neural-Kalman filter that combines the precision of NI sequence learning with the accuracy of GNSS updates, thereby reducing over-reliance on GNSS.

II. Related Work

A. Inertial Odometry Techniques

MEMS IMUs suffer from the curse of drift due to angular random walk, bias instability, and white noise [38][39]. The position estimation error is constrained by two methods:

* Yayun and Swapnil contributed equally to this work
* Contact: duyayun1@g.ucla.edu, swapnilsayan@g.ucla.edu
[1]Department of Mechanical & Aerospace Engineering, University of California, Los Angeles, 420 Westwood Plaza, Los Angeles, CA 90095
[2]Department of Electrical & Computer Engineering, University of California, Los Angeles, 420 Westwood Plaza, Los Angeles, CA 90095
[3]Amazon, 2205 7th Ave., Seattle, WA 98121 (work unrelated to Amazon)
[4]STMicroelectronics, 2755 Great America Way, Santa Clara, CA 95054

Special thanks to Matt Conroy from GoodFarms and Andre Biscaro from UCANR for their field preparation. This work was supported in part by the CONIX Research Center, one of six centers in JUMP, a Semiconductor Research Corporation (SRC) program sponsored by DARPA; by the IoBT REIGN Collaborative Research Alliance funded by the Army Research Laboratory (ARL) under Cooperative Agreement W911NF-17-2-0196; by the Air Force Office of Scientific Research (AFOSR) under Cooperative Agreement FA9550-22-1-0193; by the National Science Foundation (NSF) under awards CNS-1705135, CNS-1822935, IIS-1925360, CNS-2213839, and CMMI-2047663; by the King Abdullah University of Science and Technology (KAUST) through its Sensor Innovation research program; and by the National Institute of Food and Agriculture, USDA under awards 2021-67022-342000 and 2021-67022-34200.

TABLE I: Inertial navigation datasets

Application	Dataset
Pedestrian dead reckoning	OxIOD [23], RoNIN [20], TUM-VI [24], RIDI [25]
Autonomous driving	KITTI [26], Berkeley DeepDrive [27], Oxford RobotCar [28], nuScenes [29], Waymo Open [30]
Quadrotor tracking	Zurich Urban [31], EuRoC MAV [32]
Underwater navigation	AQUALOC [33]

Heuristic techniques: These methods use physics-based kinematic models to estimate heading and displacement. Heading estimation techniques include heuristic drift reduction, opportunistic calibration, magnetic map matching, and quasi-static moment detection [17][40]. Belief-based velocity and transportation mode constraints, such as zero-velocity updates, stride length estimation based on physiological knowledge, map information, and time-frequency analysis are used to mitigate double-integration displacement errors [41][42][43]. Kalman filters are widely used to fuse information from GPS, LIDAR, WiFi, and cameras with INS [44]. While heuristics-based techniques are computationally tractable, these "approximate" system models fail even when the assumed conditions change slightly [17][19].
Deep-learning (DL) techniques: DL-based methods are capable of capturing nonlinear motion dynamics without requiring human knowledge [17]. Techniques include using neural networks to adapt noise models and supply velocity pseudo-measurements to Kalman filters [45][46], adapt belief-based velocity constraints [47], and regress velocity, heading or displacements end to end [17][19][23][20][21]. The last technique, known as NI localization [18], is the favored approach due to its superior long-term resolution compared to heuristic methods. However, only a subset of NI localization models can be deployed on actual hardware. Furthermore, pre-trained neural networks need labeled data to fine-tune in the target domain [17], and their application potential for PA is unexplored.

B. Inertial Navigation Datasets

Table I shows several public inertial odometry datasets for localization applications in GPS-denied environments. While the aforementioned datasets are useful for prototyping NI models, Saha *et al.* [17] showed that odometry models trained on one dataset are not directly transferable to other datasets or real-world applications. The pre-trained models must be fine-tuned on a few minutes of labeled data in the target application for the model to adapt to the motion primitives. The motion dynamics of agricultural robots significantly differ from those of cars or quadrotors, as evidenced by the occurrence of wheel slippage, uneven terrain (loose nonholonomic constraints), high-amplitude motor vibrations, and constrained travel paths [9]. These make the direct usage of existing datasets for fine-tuning pre-trained models for PA non-viable. Consequently, no public inertial navigation datasets exist for PA applications.

C. Fusion of INS and GNSS

Fusing GNSS and INS using Kalman filters with heuristic propagation and update models has been extensively explored and implemented commercially, for decades [48][16][49][50][51]. The INS runs faster than GPS measurement updates, allowing agile navigation of fast-moving objects such as quadrotors [52]. The drift in INS trajectory is intermittently corrected by GNSS measurements [50], while the INS takes full control during GPS outage (e.g., inside tunnels). As we will showcase, however, the inadequate long-term odometric resolution of the heuristic-based INS model leads to an overreliance on GPS updates. While this is acceptable for applications with map information and position resolution of $\pm$1m, the approach is unsuitable for agricultural robots requiring odometric resolution as low as $\pm$10cm. The superior performance of NI navigation over heuristic-based techniques makes learning-enabled propagation models viable [20][19][45], but no prior work showcased the fusion of NI models with GPS updates.

Fig. 1: Data collection setup for the Agrobot dataset. Dotted red insets show the robot, dotted yellow insets show the ground truth setup, and the solid blue insets show the reference landmarks. (a) Phase 1 (b) Phase 2 (c) Phase 3 (d) The robot and the reference landmarks (checkerboard patterns) as viewed from the quadrotor camera in Phase 3.

III. The AgroBot Dataset

The Agrobot dataset contains 6.5 hours of 9DoF IMU data (2 hours of GPS data) and ground truth position, collected in 3 phases from an agricultural robot intended for precision farming [9][53]. We also provide an automated video processing pipeline for farmers to collect labeled ground truth position data using quadrotors for fine-tuning pre-trained NI odometry models for their own applications.

TABLE II: The AgroBot dataset

Phase	Location	IMU Model	Ground Truth Setup	Position Resolution	Length (hrs)	Distance (km)	Sequences
1	Indoor	InvenSense MPU-9250 [34]	OptiTrack 13W-P MoCap [35]	$\sim$ 0.5mm@120Hz	3.0	2.5 ($\sim$278 m / seq.)	9
2*	Rooftop farm	Bosch BNO055 [36]	OptiTrack 13W-P MoCap [35]	$\sim$ 4.0mm@60Hz	2.0	1.4 ($\sim$285 m / seq.)	5
3	Strawberry farm	Bosch BNO055 [36]	DJI Mini 2 FC7303 [37]	$\sim$ 5.0cm@30Hz	1.5	0.6 ($\sim$114 m / seq.)	6

* also has GPS data from UBlox ZED-F9R.

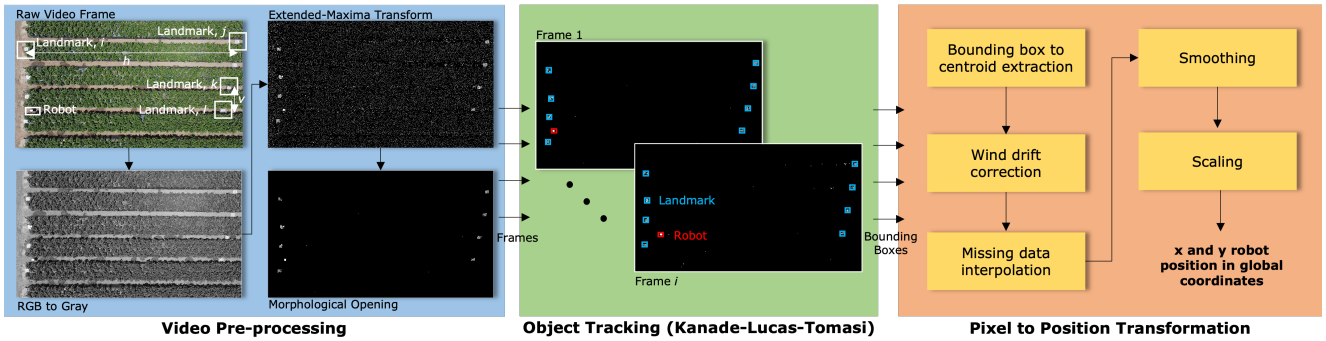

Fig. 2: Automated pipeline to extract 2D ground truth robot position in global coordinates from the quadrotor video feed.

A. Data Collection Setup

Fig. 1 and Table II illustrate the data collection setup for three phases. In all phases, we drove the agricultural robot with a remote (Logitech F710 wireless gamepad). The IMU data were logged at $\sim$100 Hz and the GPS data were logged at 1 Hz. At the beginning of each sequence, ground truth position and IMU/GNSS data were synced using "*static-rotate-static*" motion patterns. In **phase 1**, we mounted a standalone Sparkfun Razor IMU board (containing an InvenSense MPU-9250) on the robot to log the robot's dynamics on an SD card, and 6 MoCap infrared markers with 5 OptiTrack Prime 13W [35] cameras to log sub-mm resolution ground truth position on a workstation. The robot was confined to travel in an indoor space of $\sim$ 3m $\times$ 2m. Nine sequences were collected in this phase, totaling 2.5 km traveled and three hours of data. We used the ground plane supplied by OptiTrack as a reference landmark for the MoCap system. In **phase 2**, we used an in-robot Bosch BNO055 IMU [36] and UBlox ZED-F9R GNSS module to log inertial and GPS data, respectively, on the robot's control computer using ROS backend [54]. We used the same ground truth setup from phase 1 in a rooftop farm measuring $\sim$ 3.7m $\times$ 2.5m. The robot was allowed to drive between flaxseed crop lines spaced 0.3 meters apart, emulating typical flax and canola crop line spacing from North Dakota for which the robot was designed [9]. Moreover, the farm was deliberately made bumpy (loose nonholonomic constraints) and slippery (possibility of wheel slippage) for high-fidelity data collection. Across 5 sequences, 2.0 hours of data were recorded, totaling 1.4 km of distance traversed. Finally, in **phase 3**, we logged 1.5 hours of data across 6 sequences (0.6 km) on a strawberry farm in Oxnard, California. We used a DJI Mini 2 ($449) retrofitted with a DJI FC7303 gimbal-stabilized camera [37] pointing downwards to capture the robot's movements in the strawberry crop line spacings. The quadrotor is low-cost and lightweight, yet it provides a high-resolution (4k), 3-axis stabilized video feed under windy conditions (at 25 mph). The quadrotor was set to hover at $\sim$ 20m. For reference landmarks, we placed 4×8 checkerboard patterns measuring 8.5×11 inches at the edges of the camera's field of view. The robot moved in a $20\text{m} \times 8\text{m}$ area with sandy loam soil, muddy patches, uneven terrain, and 0.3 meters crop line spacing. Phase 3 environment is the most realistic in which the robot is expected to operate.

B. Video Processing Toolbox

While the MoCap system achieves sub-mm ground truth position resolution, it is not ideal for field data collection due to its limited coverage, high cost ($\sim$ $\$10^4$), reliance on specialized software and servers, and optical glare sensitivity [55]. Farmers are more likely to collect labeled data using the setup depicted in Phase 3. However, the camera' video feed does not provide position information directly. Thus, we designed a computer-executable automated pipeline for extracting 2D position (at $\pm$5.0cm resolution) from raw videos, shown in Fig. 2. The user only supplies the horizontal distance, h between landmarks i and j, and the vertical distance, v between landmarks k and l in the global coordinates to determine the scale factor of the camera. The pipeline consists of 3 steps:

Video Pre-processing: RGB video frames are first converted to grayscale. Extended maxima transform [56] then returns pixel groups with a specific intensity surrounded by pixels with a lower intensity, removing unwanted artifacts, e.g., plants, crop line spacings. Finally, morphological opening [57] with disc structuring element removes graininess, leaving only the landmarks and the robot intact.

Object Tracking: The user marks the bounding boxes for the landmarks and the robot in the first frame. The Kanade-Lucas-Tomasi tracker [58] then traces the landmarks and the robots across subsequent pre-processed video frames. We use the minimum eigenvalue algorithm [59] to extract the tracking features. In each frame, the outputs are the bounding boxes of the landmarks and the robot. Only the selected

landmarks i, j, k, and l need to stay in frame through the entire run, while others can be missed.

Pixel to Position Transformation: Pixel centroids are derived from the corner points of the bounding box. After the first frame, the robot's centroid position is corrected based on the average change of landmarks in all succeeding frames. This compensates for the quadrotor's movement caused by wind. Occasionally, the tracker may lose the robot or some of the landmarks after tracking for specific periods due to shadows or sudden pixel intensity changes. During these periods, tracking stops, and the user is prompted to perform a warm-start by re-drawing the robot and landmark bounding boxes in the first of the succeeding frames where the landmarks and the robot are noticeably visible. Linear interpolation is used to fill the gaps between warm-starts and the last known object position. Median filtering [60] smooths out high-frequency noise from the tracker. Finally, the pixel positions are converted to global coordinates by multiplying them with the scale factor s_x, s_y:

$$s_x = \frac{|C^i_{x,1} - C^j_{x,1}|}{h}, \quad s_y = \frac{|C^k_{y,1} - C^l_{y,1}|}{v}, \quad s_x \approx s_y. \quad (1)$$

$C^c_{a,b}$ is the centroid of landmark c at frame b for axis a.

IV. Neural-Kalman GNSS/INS Fusion

We propose a neural-Kalman formulation to fuse GNSS with INS. We use velocity, magnetometer, and physics-centric NI sequence learning [17] for Kalman propagation, and GNSS measurements for Kalman updates. The proposed method combines the odometric resolution of neural networks [18] with the long-term accuracy of GPS.

A. Robust Sequence Learning Formulation

The heading-displacement inertial sequence learning formulation, proposed by IONet [19], is given as follows:

$$(\Delta l_k, \Delta \psi_k) = y_\theta(\mathbf{v}^I(0), \mathbf{g}^I_0, \hat{\mathbf{a}}^I_{q:q+n}, \hat{\mathbf{w}}^I_{q:q+n}). \quad (2)$$

Under loose nonholonomic constraints, a neural network y_θ predicts the displacement Δl_t and heading rates $\Delta \psi_t$ in polar coordinates from a window of accelerometer $\hat{\mathbf{a}}^I_{q:q+n}$ and gyroscope $\hat{\mathbf{w}}^I_{q:q+n}$ readings of size n in the body frame I. The initial velocity $\mathbf{v}^I(0)$ and gravity $\mathbf{g}^I_0$ in each window are treated as latent states. Saha *et al.* [17] showed that the Eq. 2 is affected by gravity pollution, high-frequency inertial signatures, varying sensor orientation, and heading rate singularity. Thus, we use magnetometer, physics, and velocity-centric sequence learning resistant to the artifacts [17]:

$$(v_{x,k}, v_{y,k}) = y_\theta\big(\mathbf{v}^I(0), \mathbf{g}^I_0, \mathbf{N}^I_0, \hat{\mathbf{a}}^I_{q:q+n}, \hat{\mathbf{w}}^I_{q:q+n}, \hat{\mathbf{m}}^I_{q:q+n}, c_k({}^I\hat{\mathbf{a}})\big), \quad c_k({}^I\hat{\mathbf{a}}) = \left|\overline{|\mathrm{FFT}(|\hat{\mathbf{a}}^I_{q:q+n}|)|}\right| \quad (3)$$

Firstly, y_θ now predicts the 2D Cartesian velocities as opposed to the displacement and heading rates, and is robust to singularities in heading rates caused by motion primitives dominated by rotational artifacts. *Secondly*, magnetometer readings $\hat{\mathbf{m}}^I_{q:q+n}$ provide an additional global anchor $\mathbf{N}^I_0$ (the 3D magnetic north) besides $\mathbf{g}^I_0$. This makes y_θ robust to varying sensor orientation, gravity pollution, and continuous movements. *Lastly*, to prevent false triggering of y_θ due to varying sensor placement, noise, and rotational motion, y_θ is fed the absolute value of the mean discrete Fourier transform coefficients of the accelerometer norm, signifying the transportation mode of the object (e.g., static, accelerating, decelerating, constant velocity, etc.). This is known as the physics metadata channel c [17]. For window length n and stride s, the 2D position L at epoch k is:

$$\begin{cases} L_{x,k} = L_{x,k-1} + \frac{s \cdot v_{x,k}}{n-s}, \\ L_{y,k} = L_{y,k-1} + \frac{s \cdot v_{y,k}}{n-s}. \end{cases} \quad (4)$$

The network parameters θ are optimized using the strided velocity loss [20]:

$$\mathcal{L}_y = \mathbb{E}[(v_{x,g,k} - v_{x,k})^2] + \kappa \mathbb{E}[(v_{y,g,k} - v_{y,k})^2]. \quad (5)$$

B. Neural-Kalman Fusion Formulation

The discrete-time extended Kalman filter (EKF) [63] [64][65] **propagate** and **update** steps are:

$$\hat{\mathbf{x}}_{k+1|k} = f(\hat{\mathbf{x}}_k, \mathbf{u}_{k+1}, \mathbf{w}_{k+1}),$$

$$\mathbf{P}_{k+1|k} = \mathbf{F}_{k+1}\mathbf{P}_k\mathbf{F}^T_{k+1} + \mathbf{G}_{k+1}\mathbf{Q}_k\mathbf{G}^T_{k+1},$$

$$\mathbf{F}_{k+1} = \frac{\partial f}{\partial x}\Big|_{\hat{\mathbf{x}}_k, \mathbf{u}_{k+1}, \mathbf{w}_{k+1}}, \quad \mathbf{G}_{k+1} = \frac{\partial f}{\partial w}\Big|_{\hat{\mathbf{x}}_k, \mathbf{u}_{k+1}, \mathbf{w}_{k+1}}, \quad (6)$$

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1|k}\mathbf{H}^T_{k+1}\left(\underbrace{\mathbf{H}_{k+1}\mathbf{P}_{k+1|k}\mathbf{H}^T_{k+1} + \mathbf{R}_{k+1}}_{\text{innovation covariance}}\right)^{-1},$$

$$\hat{\mathbf{x}}_{k+1|k+1} = \hat{\mathbf{x}}_{k+1|k} + \mathbf{K}_{k+1}\left(\underbrace{\mathbf{z}_{k+1} - h(\hat{\mathbf{x}}_{k+1|k}, \mathbf{v}_k)}_{\text{measurement residual}}\right),$$

$$\mathbf{P}_{k+1|k+1} = (\mathbf{I} - \mathbf{K}_{k+1}\mathbf{H}_{k+1})\,\mathbf{P}_{k+1|k}, \quad \mathbf{H}_{k+1} = \frac{\partial h}{\partial x}\Big|_{\hat{\mathbf{x}}_{k+1|k}}. \quad (7)$$

In **propagate** state, the predicted state $\hat{\mathbf{x}}$ at the current epoch $k+1$ is a non-linear function f of the past state, the current control input $\mathbf{u}$, and the additive white Gaussian process noise $\mathbf{w}$ with covariance $\mathbf{Q}$. The predicted process covariance $\hat{\mathbf{P}}$, given by the Lyapunov equation [66], is computed using the Jacobians of f w.r.t. $\hat{\mathbf{x}}$ and $\mathbf{w}$ at k, linearizing the process model about current states. During **update** state based on measurements, the near-optimal Kalman gain $\mathbf{K}$ is computed by linearizing the observation model h. The measurement $\mathbf{z}$ is mixed with additive white Gaussian noise $\mathbf{v}$ with covariance $\mathbf{R}$. The measurement residual is multiplied by $\mathbf{K}$ and added to the predicted state to yield the updated state. The process covariance is updated using algebraic Riccati recursion [66].

Consider a dynamical system such that $T_\mathbf{A} : \hat{\mathbf{x}}_{k+1|k} \rightarrow \hat{\mathbf{x}}_k \mid T_\mathbf{A}$ is linear, and $g : \hat{\mathbf{x}}_{k+1|k} \rightarrow \mathbf{u}_{k+1} \mid g$ is non-linear. If $T_\mathbf{A}$ and g are linearly separable, the system evolution is:

$$\hat{\mathbf{x}}_{k+1|k} = \mathbf{A}\hat{\mathbf{x}}_k + g(\mathbf{u}_{k+1}),$$

$$\mathbf{P}_{k+1|k} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{B}_{k+1}\mathbf{U}_k\mathbf{B}^T_{k+1}, \quad \mathbf{B}_{k+1} = \frac{\partial g}{\partial u}\Big|_{\hat{\mathbf{x}}_k, \mathbf{u}_{k+1}}. \quad (8)$$

TABLE III: (Left) Position resolution and model size of our method (Neurl-KF) versus state-of-the-art baselines. (Right) Abalation study showing the effect of velocity (V), physics (P) and magnetometer (M) on the sequence learning formulation.

Method	Phase 1		Phase 2		Phase 3		Model Size (MB)*
	ATE (m)	RTE (m)	ATE (m)	RTE (m)	ATE (m)	RTE (m)	
IONet [19]	3.08|16.5	0.25|1.12	7.33|**8.43**	1.07|**0.43**	6.34|**5.44**	1.44|**0.17**	1.71
L-IONet [23]	4.06|16.4	0.13|1.17	15.9|25.7	1.05|2.05	4.37|13.7	1.55|0.97	**0.55**
AbolDeepIO [22]	5.25|15.7	0.43|1.06	12.9|21.0	0.81|0.56	3.57|25.0	1.65|1.16	12.5
VeTorch [21]	1.93|14.8	**0.11**|**0.99**	3.83|15.2	**0.26**|1.30	2.84|16.8	0.95|0.24	29.6
Neurl-KF (no GPS)	**1.66**|**7.85**	0.13|1.10	**2.72**|10.5	0.44|0.97	**0.89**|9.02	**0.26**|2.58	1.10

Method	Dataset	ATE (m)	RTE (m)	Model Size (MB)
RoNIN [20]	RoNIN [20]	4.73	1.21	6.5
TLIO [46]	TLIO [46]	2.5	0.15	∼6.5
AI-IMU DR [45]	KITTI [26]	16.8	1.10	0.67
Neurl-KF (no GPS)	AgroBot	1.76|9.12	0.28|1.55	1.10

Method@	Phase 1		Phase 2		Phase 3		Code Size (MB)
	ATE (m)	RTE (m)	ATE (m)	RTE (m)	ATE (m)	RTE (m)	
UKF-M INS+GPS [61]	4.06	**0.18**	4.35	**0.21**	8.09	1.07	0.192
EKF INS+GPS [62]	2.22	0.35	2.24	0.35	5.48	1.05	**0.077**
GPS only (no INS)	1.90	0.40	**1.88**	0.40	**1.89**	0.45	-
Neurl-KF (with GPS)	**0.37**|**1.07**	**0.16**|0.30	**1.08**|2.20	0.41|1.16	**1.63**|2.17	**0.28**|**0.45**	1.12

V	P	M	Pha. 1 ATE (m)
✓	✓	✓	**1.66**|**7.85**
✓	✓	✗	2.04|29.8
✓	✗	✓	2.26|9.13
✗	✓	✓	2.53|16.4
✓	✗	✗	1.86 |21.0
✗	✓	✗	2.00 |16.3
✗	✗	✓	3.22|16.3
✗	✗	✗	2.46 |16.6

* unfrozen (.hdf5) model size; @ GPS rate: 1 Hz
first term in ATE/RTE is on seen trajectory, second term is on unseen trajectory; single term is on unseen trajectory
best method, second best method

For physics, velocity, and magnetometer-centric NI navigation, under loose nonholonomic constraints, the Kalman propagation is given as:

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{L}_x \\ \hat{L}_y \\ v_x \\ v_y \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \mathbf{a}^I_{q:q+n} \\ \mathbf{w}^I_{q:q+n} \\ \mathbf{m}^I_{q:q+n} \\ c(\mathbf{a}^I_{q:q+n}) \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} \mathbf{I}_{2\times 2} & \mathbf{0}_{2\times 2} \\ \mathbf{0}_{2\times 2} & \mathbf{0}_{2\times 2} \end{bmatrix}, \quad (9)$$

$$g(\cdot) = \begin{bmatrix} \Delta t \cdot \mathbf{I}_{2\times 2} \\ \mathbf{I}_{2\times 2} \end{bmatrix} \cdot y_\theta(\cdot), \quad \Delta t = \frac{s}{n-s}. \quad (10)$$

The neural network $y_\theta(\cdot)$ provides a non-linear black-box mapping between $\mathbf{u}$ and $\hat{\mathbf{x}}$, supplying velocity pseudo-measurements from raw IMU readings. Thus, $\mathbf{B}_{k+1}$ is the linearized Jacobian of $y_\theta(\cdot)$ w.r.t. the 9-DoF IMU readings and the physics channel in the current input window:

$$\mathbf{B}_{k+1} = \begin{bmatrix} \frac{\Delta t \partial y_{\theta_x}}{\partial \mathbf{a}^I_{q:q+n}} & \frac{\Delta t \partial y_{\theta_x}}{\partial \mathbf{w}^I_{q:q+n}} & \frac{\Delta t \partial y_{\theta_x}}{\partial \mathbf{m}^I_{q:q+n}} & \frac{\Delta t \partial y_{\theta_x}}{\partial c(\mathbf{a}^I_{q:q+n})} \\ \frac{\Delta t \partial y_{\theta_y}}{\partial \mathbf{a}^I_{q:q+n}} & \frac{\Delta t \partial y_{\theta_y}}{\partial \mathbf{w}^I_{q:q+n}} & \frac{\Delta t \partial y_{\theta_y}}{\partial \mathbf{m}^I_{q:q+n}} & \frac{\Delta t \partial y_{\theta_y}}{\partial c(\mathbf{a}^I_{q:q+n})} \\ \frac{\partial y_{\theta_x}}{\partial \mathbf{a}^I_{q:q+n}} & \frac{\partial y_{\theta_x}}{\partial \mathbf{w}^I_{q:q+n}} & \frac{\partial y_{\theta_x}}{\partial \mathbf{m}^I_{q:q+n}} & \frac{\partial y_{\theta_x}}{\partial c(\mathbf{a}^I_{q:q+n})} \\ \frac{\partial y_{\theta_y}}{\partial \mathbf{a}^I_{q:q+n}} & \frac{\partial y_{\theta_y}}{\partial \mathbf{w}^I_{q:q+n}} & \frac{\partial y_{\theta_y}}{\partial \mathbf{m}^I_{q:q+n}} & \frac{\partial y_{\theta_y}}{\partial c(\mathbf{a}^I_{q:q+n})} \end{bmatrix}. \quad (11)$$

$\mathbf{U}$ consists of Allan variance parameters [67] of the IMU, including the accelerometer noise variance ($\sigma^2(n_{\mathbf{a}^I})$), variance in gyroscope angular random walk ($\sigma^2(\mathbf{w}^I_{\text{ARW}}\sqrt{\Delta t})$) and bias instability ($\sigma^2(\mathbf{w}^I_{\text{BI}})$), gyroscope noise variance ($\sigma^2(n_{\mathbf{w}^I})$), and the magnetometer noise variance $\sigma^2(n_{\mathbf{m}^I})$. The following $\mathbf{U}$ also contains the sum of the accelerometer noise variances in each axis owing to c:

$$\mathbf{U} = \text{diag}\big(\sigma^2(n_{\mathbf{a}^I}), \sigma^2(\mathbf{w}^I_{\text{ARW}}\sqrt{\Delta t}) + \sigma^2(\mathbf{w}^I_{\text{BI}}) + \sigma^2(n_{\mathbf{w}^I}), \sigma^2(n_{\mathbf{m}^I}), \sum \sigma^2(n_{\mathbf{a}^I})\big). \quad (12)$$

The measurement update $\mathbf{z}$ comes from the GNSS module. h denotes the inverse mapping from longitude-latitude to 2D Cartesian coordinates and the resulting 2D Cartesian velocities with the WGS-84 ellipsoid geodetic model of the Earth [68]. $\mathbf{R}$ is the covariance of GNSS position estimation error and velocity estimation error.

C. Backbone Neural Architecture

To extract spatial and temporal features jointly and hierarchically, we use a temporal convolutional network (TCN) [69][70] to model y_θ [45][23][21]. *Firstly*, the use of dilated convolution kernels allows TCNs to have a larger receptive field with lower layer count, memory footprint, and parameters than CNNs, allowing y_θ to discover global context in long sequences at high input resolution. *Secondly*, TCNs use causal convolution to maintain temporal ordering without using computationally-expensive recurrent units, which suffer from the exploding and vanishing gradient problem [71]. *Thirdly*, TCNs fuse layers using gated residual blocks for non-linear temporal interaction modeling [23].

V. Evaluation

We used 80% of the AgroBot dataset for training the NI models and 20% for testing (split by sequence). The window size, n was 100 (1 second), and the stride, s was 20 (0.2 second). The backbone TCN had 8 layers (dilation factors: 1,2,4,8,16,32,64,128), 32 filters, a kernel size of 5, and skip connections. The outputs of the TCN were reshaped, pooled, flattened, and fed to a 32-unit dense layer. The batch size, optimizer, and training epochs were set to 256, Adam (learning rate: 0.001), and 3000, respectively. The models were implemented in Keras [72] using TensorFlow backend [73] on a Lambda GPU workstation. The final models were deployed on the robot's Nvidia Jetson Nano board, featuring 1.43 GHz ARM A57 CPU, 128-core Maxwell GPU, and 4-GB RAM [74]. The NN baselines [19], [21], [22], [23] were implemented by us and trained fresh on the AgroBot dataset.

A. Localization Performance and Resource Usage

Table III (left) summarizes the absolute trajectory error (ATE, root-mean-squared-error between predicted and ground truth locations for the whole sequence) [20] and relative trajectory error (RTE, ATE over 1 minute intervals) [20] of our model (Neurl-KF) and competing proposals

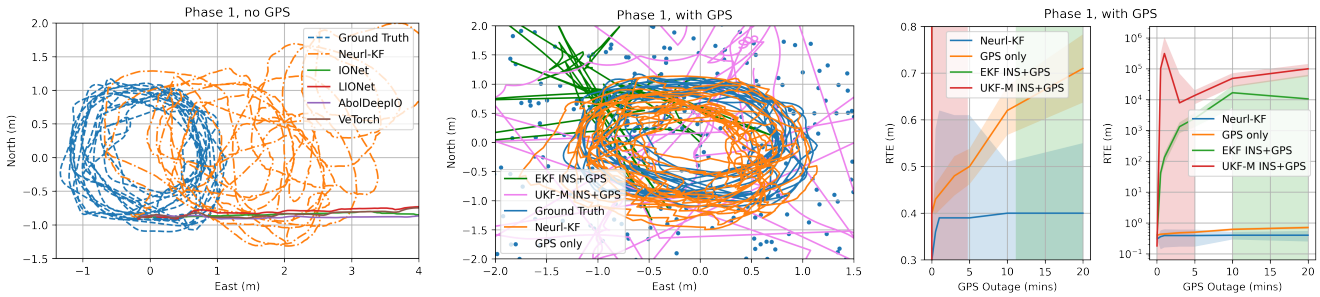

Fig. 3: (Left) Sample trajectory reconstruction (no GPS, traveling 100 m) for Neurl-KF and competing proposals. (Center) Sample trajectory reconstruction (1 Hz GPS, traveling 100 m) for Neurl-KF and competing proposals. (Right) Error evolution (RTE) versus GPS outage time of Neurl-KF and competing proposals. The left graph is a zoomed-in part of the right graph.

on the AgroBot dataset. Compared to competing NI models (no GPS) [19][22][21][20], Neurl-KF lowers code size and ATE by 1.5 - 27× and 1.4 - 5.8×, respectively while having comparable RTE, tracking the robot within 1.4 m. The lightweight neural backbone, combined with robust sequence learning constrains both the error and the model size. Compared to classical GPS-enabled techniques [61][62], at 1 Hz GNSS updates, Neurl-KF lowers the ATE and RTE by 1.2 - 11× and 1.1 - 3.8×, respectively. Fig. 3 (left, center) shows sample trajectory plots for Neurl-KF and baselines. Competing NI models are smooth but drift rapidly, unable to handle wheel slippage, rotational artifacts, uneven terrain, and motor vibrations. GPS-enabled baselines trust the noisy GPS updates more than the IMU, leading to a noisy trajectory. Neurl-KF, on the other hand, learns to retain the smoothness of NI odometry and the accuracy of GPS.

B. Ablation Study for Sequence Learning Formulation

Table III (right) shows the importance of each component in our velocity, magnetometer, and physics-centric sequence-learning formulation. Our formulation lowers the ATE by 1.1 - 3.8×. The addition of magnetometer data and changing the output of y_θ to regress velocity instead of $(\Delta l_k, \Delta\phi_k)$ reduces the ATE the most.

C. Error Evolution with GPS Outage

Fig. 3 (right) shows the RTE of Neurl-KF and baselines under varying GPS outage times. Our technique constrains the RTE and ATE to 0.4 m and 2.75 m, respectively, even with 20 minutes of GPS outage. Competing INS+GNSS baselines drift quickly due to IMU double integration error explosion. While the GPS-only baseline has similar ATE and RTE with our Neurl-KF, direct GNSS updates are noisy and unusable, as shown in Fig. 3 (center). Note that the sudden drop in RTE for baselines at 5 minutes and 20 minutes is likely due to the measurement updates coinciding with less noisy GNSS measurements closer to the ground truth.

D. Fine-Tuning Performance

Table IV illustrates the performance of pre-trained models on unseen datasets, as well as RTE comparison between pre-trained models and models trained from scratch. Without fine-tuning, NI models trained on one dataset are not directly transferable on another dataset, as the RTE on the unseen dataset is 1.6 - 13.6× higher than on the trained dataset. The difference in IMU noise characteristics and motion artifacts leads to high RTE in the new domain. With transfer learning on small amounts of labeled data in the target domain, the RTE in the target domain matches the in-domain RTE. In fact, fine-tuning pre-trained models in a targeted domain requires <20× data than training a model from scratch. Fine-tuning with 1 minute of labeled data outperforms training from scratch with 20 minutes of data by 1.3×.

TABLE IV: Fine-tuning performance (RTE, m)

Training Dataset	**RTE (m) on Inference Dataset (Unseen Trajectory)**					
	P1	**P1 (FT)**	**P2**	**P2 (FT)**	**P3**	**P3 (FT)**
P1	1.10	-	14.5	**1.45**	15.0	**4.96**
P2	2.71	**1.09**	0.97	-	4.25	**2.63**
P3	1.85	**0.76**	1.93	**1.15**	2.58	-

Method	**RTE (m) with T minutes of data in the new domain***		
	T = 1	T = 5	T = 20
Train from scratch	26.8	3.29	2.55
Fine-tune*	**1.92**	**1.62**	**1.45**

P1: Phase 1, P2: Phase 2, P3: Phase 3
FT: Fine-tuning with 20 minutes of data in the new domain for 100 epochs
* The pre-trained model was trained on Phase 1 data; target dataset: Phase 2

VI. Discussion and Future Work

This paper presents a dataset, ground truth data collection tools, and a robust yet lightweight neural-Kalman formulation for developing high-resolution localization solutions for precision agricultural robots. Our solution tracks robots within 1.4m with 1 Hz GNSS updates, and within 2.75 m with 20 minutes of GNSS outage. Neurl-KF balances the trust between GPS and IMU better than existing conventional and neural INS, providing 1.1-5.8× higher resolution while being resource-efficient and robust to inertial disturbances. The user-friendly vision pipeline allows pre-trained models to be adapted to the target domain with only a minute of labeled data, improving deployability in the wild. The main drawback of our solution is the need for fine-tuning pre-trained models with high-resolution labeled data in the target domain. Possible use of on-device domain adaptation, on-device learning, and use of BERT-like pre-trained IMU embeddings [75] need to be explored to automate the tuning process with unlabelled data [17].

REFERENCES

[1] R. Gebbers and V. I. Adamchuk, "Precision agriculture and food security," Science, vol. 327, no. 5967, pp. 828–831, 2010.

[2] R. Bongiovanni and J. Lowenberg-DeBoer, "Precision agriculture and sustainability," Precision agriculture, vol. 5, no. 4, pp. 359–387, 2004.

[3] J. J. Roldán, J. del Cerro, D. Garzón-Ramos, P. Garcia-Aunon, M. Garzón, J. De León, and A. Barrientos, "Robots in agriculture: State of art and practical experiences," Service robots, pp. 67–90, 2018.

[4] A. Milioto, P. Lottes, and C. Stachniss, "Real-time semantic segmentation of crop and weed for precision agriculture robots leveraging background knowledge in cnns," in 2018 IEEE international conference on robotics and automation (ICRA), pp. 2229–2235, IEEE, 2018.

[5] E. Mavridou, E. Vrochidou, G. A. Papakostas, T. Pachidis, and V. G. Kaburlasos, "Machine vision systems in precision agriculture for crop farming," Journal of Imaging, vol. 5, no. 12, p. 89, 2019.

[6] R. Bogue, "Sensors key to advances in precision agriculture," Sensor Review, 2017.

[7] N. Chebrolu, P. Lottes, T. Läbe, and C. Stachniss, "Robot localization based on aerial images for precision agriculture tasks in crop fields," in 2019 International Conference on Robotics and Automation (ICRA), pp. 1787–1793, IEEE, 2019.

[8] J. Guo, X. Li, Z. Li, L. Hu, G. Yang, C. Zhao, D. Fairbairn, D. Watson, and M. Ge, "Multi-gnss precise point positioning for precision agriculture," Precision agriculture, vol. 19, no. 5, pp. 895–911, 2018.

[9] Y. Du, B. Mallajosyula, D. Sun, J. Chen, Z. Zhao, M. Rahman, M. Quadir, and M. K. Jawed, "A low-cost robot with autonomous recharge and navigation for weed control in fields with narrow row spacing," in 2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 3263–3270, IEEE, 2021.

[10] J. Yi, H. Wang, J. Zhang, D. Song, S. Jayasuriya, and J. Liu, "Kinematic modeling and analysis of skid-steered mobile robots with applications to low-cost inertial-measurement-unit-based motion estimation," IEEE transactions on robotics, vol. 25, no. 5, pp. 1087–1097, 2009.

[11] Y. Yan, B. Zhang, J. Zhou, Y. Zhang, X. Liu, et al., "Real-time localization and mapping utilizing multi-sensor fusion and visual–imu–wheel odometry for agricultural robots in unstructured, dynamic and gps-denied greenhouse environments," Agronomy, vol. 12, no. 8, p. 1740, 2022.

[12] M. Brossard, A. Barrau, and S. Bonnabel, "Rins-w: Robust inertial navigation system on wheels," in 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2068–2075, IEEE, 2019.

[13] P. Tokekar, J. Vander Hook, D. Mulla, and V. Isler, "Sensor planning for a symbiotic uav and ugv system for precision agriculture," IEEE Transactions on Robotics, vol. 32, no. 6, pp. 1498–1511, 2016.

[14] J. Das, G. Cross, C. Qu, A. Makineni, P. Tokekar, Y. Mulgaonkar, and V. Kumar, "Devices, systems, and methods for automated monitoring enabling precision agriculture," in 2015 IEEE International Conference on Automation Science and Engineering (CASE), pp. 462–469, IEEE, 2015.

[15] Y. Zhang, F. Gao, and L. Tian, "Ins/gps integrated navigation for wheeled agricultural robot based on sigma-point kalman filter," in 2008 Asia Simulation Conference-7th International Conference on System Simulation and Scientific Computing, pp. 1425–1431, IEEE, 2008.

[16] S. Sukkarieh, E. M. Nebot, and H. F. Durrant-Whyte, "A high integrity imu/gps navigation loop for autonomous land vehicle applications," IEEE transactions on robotics and automation, vol. 15, no. 3, pp. 572–578, 1999.

[17] S. S. Saha, S. S. Sandha, L. A. Garcia, and M. Srivastava, "Tinyodom: Hardware-aware efficient neural inertial navigation," Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, vol. 6, no. 2, pp. 1–32, 2022.

[18] S. Herath, D. Caruso, C. Liu, Y. Chen, and Y. Furukawa, "Neural inertial localization," in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6604–6613, 2022.

[19] C. Chen, X. Lu, A. Markham, and N. Trigoni, "Ionet: Learning to cure the curse of drift in inertial odometry," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, 2018.

[20] S. Herath, H. Yan, and Y. Furukawa, "Ronin: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods," in 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 3146–3152, IEEE, 2020.

[21] R. Gao, X. Xiao, S. Zhu, W. Xing, C. Li, L. Liu, L. Ma, and H. Chai, "Glow in the dark: Smartphone inertial odometry for vehicle tracking in gps blocked environments," IEEE Internet of Things Journal, vol. 8, no. 16, pp. 12955–12967, 2021.

[22] M. A. Esfahani, H. Wang, K. Wu, and S. Yuan, "Aboldeepio: A novel deep inertial odometry network for autonomous vehicles," IEEE Transactions on Intelligent Transportation Systems, vol. 21, no. 5, pp. 1941–1950, 2019.

[23] C. Chen, P. Zhao, C. X. Lu, W. Wang, A. Markham, and N. Trigoni, "Deep-learning-based pedestrian inertial navigation: Methods, data set, and on-device inference," IEEE Internet of Things Journal, vol. 7, no. 5, pp. 4431–4441, 2020.

[24] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stückler, and D. Cremers, "The tum vi benchmark for evaluating visual-inertial odometry," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1680–1687, IEEE, 2018.

[25] H. Yan, Q. Shan, and Y. Furukawa, "Ridi: Robust imu double integration," in Proceedings of the European Conference on Computer Vision (ECCV), pp. 621–636, 2018.

[26] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," The International Journal of Robotics Research, vol. 32, no. 11, pp. 1231–1237, 2013.

[27] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," in Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2174–2182, 2017.

[28] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," The International Journal of Robotics Research, vol. 36, no. 1, pp. 3–15, 2017.

[29] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 11621–11631, 2020.

[30] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al., "Scalability in perception for autonomous driving: Waymo open dataset," in Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 2446–2454, 2020.

[31] A. L. Majdik, C. Till, and D. Scaramuzza, "The zurich urban micro aerial vehicle dataset," The International Journal of Robotics Research, vol. 36, no. 3, pp. 269–273, 2017.

[32] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," The International Journal of Robotics Research, vol. 35, no. 10, pp. 1157–1163, 2016.

[33] M. Ferrera, V. Creuze, J. Moras, and P. Trouvé-Peloux, "Aqualoc: An underwater dataset for visual–inertial–pressure localization," The International Journal of Robotics Research, vol. 38, no. 14, pp. 1549–1559, 2019.

[34] J. Zheng, M. Qi, K. Xiang, and M. Pang, "Imu performance analysis for a pedestrian tracker," in International Conference on Intelligent Robotics and Applications, pp. 494–504, Springer, 2017.

[35] J. S. Furtado, H. H. Liu, G. Lai, H. Lacheray, and J. Desouza-Coelho, "Comparative analysis of optitrack motion capture systems," in Advances in Motion Sensing and Control for Robotic Applications, pp. 15–31, Springer, 2019.

[36] N. Piotrowski, "Implementation and characterization of commercial off-the-shelf inertial measurement units for the attitude determination system of the move-iii cubesat," in 36th Annual Small Satellite Conference, 2022.

[37] M. Stanković, M. M. Mirza, and U. Karabiyik, "Uav forensics: Dji mini 2 case study," Drones, vol. 5, no. 2, p. 49, 2021.

[38] M. Kok, J. Hol, and T. Schön, "Using inertial sensors for position and orientation estimation," Foundations and Trends in Signal Processing, vol. 11, pp. 1–153, 2017.

[39] S. Shen, M. Gowda, and R. Roy Choudhury, "Closing the gaps in inertial motion tracking," in Proceedings of the 24th Annual International Conference on Mobile Computing and Networking, pp. 429–444, 2018.

[40] Y. Wu, H.-B. Zhu, Q.-X. Du, and S.-M. Tang, "A survey of the research status of pedestrian dead reckoning systems based on inertial sensors," International Journal of Automation and Computing, vol. 16, no. 1, pp. 65–83, 2019.

[41] R. Harle, "A survey of indoor inertial positioning systems for pedestrians," IEEE Communications Surveys & Tutorials, vol. 15, no. 3, pp. 1281–1293, 2013.

[42] A. R. Jiménez, F. Seco, J. C. Prieto, and J. Guevara, "Indoor pedestrian navigation using an ins/ekf framework for yaw drift reduction and a foot-mounted imu," in 2010 7th workshop on positioning, navigation and communication, pp. 135–143, IEEE, 2010.

[43] X. Hou and J. Bergmann, "Pedestrian dead reckoning with wearable sensors: A systematic review," IEEE Sensors Journal, vol. 21, no. 1, pp. 143–152, 2020.

[44] S. Kuutti, S. Fallah, K. Katsaros, M. Dianati, F. Mccullough, and A. Mouzakitis, "A survey of the state-of-the-art localization techniques and their potentials for autonomous vehicle applications," IEEE Internet of Things Journal, vol. 5, no. 2, pp. 829–846, 2018.

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

[46] W. Liu, D. Caruso, E. Ilg, J. Dong, A. I. Mourikis, K. Daniilidis, V. Kumar, and J. Engel, "Tlio: Tight learned inertial odometry," IEEE Robotics and Automation Letters, vol. 5, no. 4, pp. 5653–5660, 2020.

[47] B. Wagstaff and J. Kelly, "Lstm-based zero-velocity detection for robust inertial navigation," in 2018 International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1–8, IEEE, 2018.

[48] F. Caron, E. Duflos, D. Pomorski, and P. Vanheeghe, "Gps/imu data fusion using multisensor kalman filtering: introduction of contextual aspects," Information fusion, vol. 7, no. 2, pp. 221–230, 2006.

[49] E. Brookner, Tracking and Kalman Filtering Made Easy. Wiley Online Library, 1998.

[50] P. D. Groves, Principles of GNSS, Inertial, and Multisensor Integrated Navigation Systems. Artech House, Fitchburg, MA, 2008.

[51] I. Puente, H. González-Jorge, J. Martínez-Sánchez, and P. Arias, "Review of mobile mapping and surveying technologies," Measurement, vol. 46, no. 7, pp. 2127–2145, 2013.

[52] J. Wendel, O. Meister, C. Schlaile, and G. F. Trommer, "An integrated gps/mems-imu navigation system for an autonomous helicopter," Aerospace science and technology, vol. 10, no. 6, pp. 527–533, 2006.

[53] Y. Du, G. Zhang, D. Tsang, and M. K. Jawed, "Deep-cnn based robotic multi-class under-canopy weed control in precision farming," in 2022 International Conference on Robotics and Automation (ICRA), pp. 2273–2279, IEEE, 2022.

[54] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, et al., "Ros: an open-source robot operating system," in ICRA workshop on open source software, vol. 3, p. 5, Kobe, Japan, 2009.

[55] D. Shao, X. Liu, B. Cheng, O. Wang, and T. Hoang, "Edge4real: A cost-effective edge computing based human behaviour recognition system for human-centric software engineering," in Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering, pp. 1287–1291, 2020.

[56] P. Soille et al., Morphological image analysis: principles and applications, vol. 2. Springer, 1999.

[57] R. Van Den Boomgaard and R. Van Balen, "Methods for fast morphological image transforms using bitmapped binary images," CVGIP: Graphical Models and Image Processing, vol. 54, no. 3, pp. 252–258, 1992.

[58] C. Tomasi and T. Kanade, "Detection and tracking of point," Int J Comput Vis, vol. 9, pp. 137–154, 1991.

[59] J. Shi and C. Tomasi, "Good features to track," in 1994 Proceedings of IEEE conference on computer vision and pattern recognition, pp. 593–600, IEEE, 1994.

[60] B. Justusson, "Median filtering: Statistical properties," Two-Dimensional Digital Signal Prcessing II, pp. 161–196, 1981.

[61] M. Brossard, S. Bonnabel, and J.-P. Condomines, "Unscented kalman filtering on lie groups," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 2485–2491, IEEE, 2017.

[62] H. Qi and J. B. Moore, "Direct kalman filtering approach for gps/ins integration," IEEE Transactions on Aerospace and Electronic Systems, vol. 38, no. 2, pp. 687–693, 2002.

[63] L. McGee, S. Schmidt, and G. Smith, "Applications of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle," NASA Technical Report R-135, Tech. Rep, 1962.

[64] H. W. Sorenson, "On the development of practical nonlinear filters," Information Sciences, vol. 7, pp. 253–270, 1974.

[65] M. I. Ribeiro, "Kalman and extended kalman filters: Concept, derivation and properties," Institute for Systems and Robotics, vol. 43, p. 46, 2004.

[66] B. Sinopoli, L. Schenato, M. Franceschetti, K. Poolla, M. I. Jordan, and S. S. Sastry, "Kalman filtering with intermittent observations," IEEE transactions on Automatic Control, vol. 49, no. 9, pp. 1453–1464, 2004.

[67] N. El-Sheimy, H. Hou, and X. Niu, "Analysis and modeling of inertial sensors using allan variance," IEEE Transactions on instrumentation and measurement, vol. 57, no. 1, pp. 140–149, 2007.

[68] M. Kumar, "World geodetic system 1984: A modern and accurate global reference frame," Marine Geodesy, vol. 12, no. 2, pp. 117–126, 1988.

[69] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu, "Wavenet: A generative model for raw audio," in 9th ISCA Speech Synthesis Workshop, 2016.

[70] C. Lea, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks: A unified approach to action segmentation," in European Conference on Computer Vision, pp. 47–54, Springer, 2016.

[71] S. S. Saha, S. S. Sandha, S. Pei, V. Jain, Z. Wang, Y. Li, A. Sarker, and M. Srivastava, "Auritus: An open-source optimization toolkit for training and development of human movement models and filters using earables," Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies, vol. 6, no. 2, pp. 1–34, 2022.

[72] A. Gulli and S. Pal, Deep learning with Keras. Packt Publishing Ltd, 2017.

[73] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al., "{TensorFlow}: a system for {Large-Scale} machine learning," in 12th USENIX symposium on operating systems design and implementation (OSDI 16), pp. 265–283, 2016.

[74] A. A. Süzen, B. Duman, and B. Şen, "Benchmark analysis of jetson tx2, jetson nano and raspberry pi using deep-cnn," in 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA), pp. 1–5, IEEE, 2020.

[75] H. Xu, P. Zhou, R. Tan, M. Li, and G. Shen, "Limu-bert: Unleashing the potential of unlabeled data for imu sensing applications," in Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems, pp. 220–233, 2021.