RoNIN: Robust Neural Inertial Navigation in the Wild: Benchmark, Evaluations, & New Methods

Sachini Herath1† Hang Yan2† Yasutaka Furukawa1

Abstract—This paper sets a new foundation for data-driven inertial navigation research, where the task is the estimation of horizontal positions and heading direction of a moving subject from a sequence of IMU sensor measurements from a phone. In contrast to existing methods, our method can handle varying phone orientations and placements.

More concretely, the paper presents 1) a new benchmark containing more than 40 hours of IMU sensor data from 100 human subjects with ground-truth 3D trajectories under natural human motions; 2) novel neural inertial navigation architectures, making significant improvements for challenging motion cases; and 3) qualitative and quantitative evaluations of the competing methods over three inertial navigation benchmarks. We share the code and data to promote further research. (http://ronin.cs.sfu.ca)

I. INTRODUCTION

An inertial measurement unit (IMU), a combination of accelerometer, gyroscope, and magnetometer, plays an important role in navigation applications. IMU sensor fusion produces real-time orientations of Virtual Reality headsets, augments SLAM [1]–[3] in Augmented Reality [4]–[6], and allows enhanced navigation and control in many domains.

Inertial navigation is the ultimate form of IMU-based navigation, whose task is to estimate positions and orientations of a moving subject from a sequence of IMU sensor measurements (see Fig. 1). IMUs are energy-efficient, work anywhere even inside pockets, and are in every smartphone, which everyone carries everyday all the time.

Most existing inertial navigation algorithms, such as zero speed update [7] and step counting [8], require unrealistic constraints that are incompatible with everyday smartphone usage scenarios. Data-driven approaches [9], [10] have recently made a breakthrough in loosening these constraints, where IMU sensor data and ground-truth motion trajectories allows supervised learning of direct motion parameters. Nevertheless, these methods only focus on regressing position for few distinct carrying types.

This paper seeks to strengthen data-driven inertial navigation research by presenting:
• the largest inertial navigation database consisting of more than 42.7 hours of IMU and ground-truth 3D motion data from 100 human subjects handling smartphones naturally as in real day-to-day activities.
• novel neural architectures making significant improvements over the existing best method for 1) position estimation and 2) body heading estimation, invariant to phone orientation.
• extensive qualitative/quantitative evaluations of existing baselines and state-of-the-art methods on three benchmarks.

We share the code and data to promote further research in a hope to establish an ultimate anytime anywhere navigation system for everyone’s smartphone.

II. RELATED WORK

We group inertial navigation algorithms into three categories based on their use of priors.

Physics-based (no priors): IMU double integration uses acceleration, with device orientation and gravity estimation (e.g., via Kalman filter [11] on IMU signals) to estimate positions. In practice, sensor biases explodes in the double integration process, and these systems fail without additional constraints. A foot mounted IMU with zero speed update is probably the most successful example, where the sensor bias can be corrected subject to a constraint that the velocity must become zero whenever a foot touches the ground [7].

Heuristic priors: The methods exploit the repetitive nature of human motions. Step counting assumes that 1) An IMU is rigidly attached to a body; 2) The motion direction is...
fixed with respect to the IMU; and 3) The distance of travel is proportional to the number of foot-steps. The method produces impressive results in a controlled environment where these assumptions are assured. More sophisticated approaches utilize principal component analysis [12] or frequency domain analysis [13] to infer motion directions.

**Data-driven priors:** Robust IMU double integration (RIDI) [9] focuses on regressing velocity vectors in a device coordinate frame, while relying on traditional sensor fusion methods to estimate device orientations. RIDI works for complex motion cases such as backward-walking, significantly improving and expanding the operating ranges of the inertial navigation system. IONet is a neural network based approach, which regresses the velocity magnitude and the rate of motion-heading change without relying on external device orientation information [14].

**Inertial navigation datasets:** Table I summarizes the two existing datasets: RIDI [9] and OXIOD [10]. The common issue in these datasets is the reliance on a single device for both IMU data and the ground-truth motion acquisition. The phone must have a clean line-of-sight for Visual Inertial SLAM or must be clearly visible for the Vicon system all the time, prohibiting natural phone handling especially for a bag and a leg pocket scenarios.

III. THE RoNIN DATASET

Scale, diversity and fidelity are the three key factors in building a next-generation inertial navigation database. In comparison to the current largest database OXIOD [10], our dataset boasts of: (see Table I)

- Scale: 2.9 times more IMU-motion data with over 276 sequences in 3 buildings,
- Diversity: 20 times more human subjects, with Android devices from three vendors1.

1 Asus Zenfone AR, Samsung Galaxy S9 and Google Pixel 2 XL. The first uses ICM20602 IMU sensor and the latter two use LSM6DSL.

- Fidelity: subjects handle devices naturally as in real day-to-day activities such as carrying inside a bag, placing deep inside a pocket, or picking up by hand, while walking, sitting or wandering around.

We have developed a two-device data acquisition protocol, where we use a harness to attach a 3D tracking phone (Asus Zenfone AR) to a body and let subjects handle the other phone freely for IMU data collection (See Fig. 2). Besides the benefits of allowing natural body motions and phone handling, this protocol exhibits two important changes to the nature of motion learning tasks.

1) The positional ground-truth is obtained only for the 3D tracking phone attached to a harness, and we estimate the trajectory of a body instead of the IMU phone.

2) The data offers a new task of body heading estimation. A standard sensor fusion algorithm works well for the device orientation estimation [15], [16]. However, the body heading is more challenging as it differs from the device orientation arbitrarily depending on how one carries a phone. We collect the ground-truth body headings by assuming that they are identical to the headings of the tracking phone attached to the body, after compensating for the constant offset introduced by the misalignment of the harness.

We have made great engineering efforts in implementing the data processing pipeline to ensure high-quality sensor data and ground-truth. Through quantitative assessments, we ensured that our “ground truth” trajectories drift less than 0.3m after 10 minutes of activities. Similarly, the device orientation from Android system API drift less than 20°, while our system further reduces it to less than 10°, which we treat as ground-truth. Both IMU sensor data and 3D pose data are recorded at 200Hz. We also record measurements from Android software-based sensors and the barometer.

We take 85% of subjects and divide their data into training, validating and testing (seen) subsets. Remaining 15% are used to test the generalization capability of the model to unseen human subjects.

IV. ROBUST NEURAL INERTIAL NAVIGATION (RoNIN)

Our neural architecture for inertial navigation, dubbed Robust Neural Inertial Navigation (RoNIN), seeks to regress a 2D vector \( \vec{y} \), given an IMU sensor history of \( k \) frames, \( X_{t+k} \), consisting of acceleration and angular velocity.

We use two key design principles: 1) Coordinate frame normalization defining the input and output feature space and 2) Robust velocity losses improving the signal-to-noise-ratio.

---

**TABLE I**

| Dataset | Device Configuration | No. of subjects | Length (hours) | Device placements |
|---------|----------------------|-----------------|----------------|-------------------|
| RIDI    | 1 device as IMU & 3D tracking phone | 10 | 2.5 | hand, pocket, bag, body |
| OXIOD  | IMU phone tracked by Vicon | 5 | 14.7 | hand, pocket, bag, trolley |
| RoNIN  | IMU phone and 3D tracking phone | 100 | 42.7 | Unrestricted |
Fig. 3. RoNIN Architectures: Neural network modules are in green and transformation layers are in white.

Fig. 4. Coordinate Systems (b) shows 2 of infinitely many possible HACFs. Y axis of (c) is taken as the heading direction.

First, we define these design principals for position estimation task, where $\vec{y}$ is the horizontal velocity of the subject. We use three well-known architectures, ResNet [17], Long Short Term Memory Network (LSTM) [18], or Temporal Convolutional Network (TCN) [19], to demonstrate that our core ideas are universal. Finally we apply the same principals to estimate body heading using the LSTM network.

A. Coordinate frame normalization

Feature representations, in our case the choice of coordinate frames (CF), have significant impacts on training. IMU sensor measurements come from moving device coordinate frames (Fig. 4a.), while ground-truth motion trajectories come from a global coordinate frame. RoNIN uses a heading-agnostic coordinate frame (HACF) (Fig. 4b) to represent both input IMU and output data.

Device CF changes every frame, thus velocity of the subject, when transformed to device CF, would change depending on how one holds a phone even for exactly the same motions. RIDI [9] proposed the stabilized IMU coordinate frame, which is obtained from the device CF by aligning its Y-axis with the gravity direction for each frame. However, this alignment process has a singularity (ambiguity) when the Y-axis points towards the gravity (e.g., a phone is inside a leg pocket upside-down), making the regression task harder, usually completely fail due to the randomness.

RoNIN uses HACF, that is, any coordinate frame whose Z axis is aligned with gravity. In other words, one can pick any such coordinate frame as long as we keep it consistent throughout the sequence. The coordinate transformation into HACF does not suffer from singularities or discontinuities with proper rotation representation, e.g. with quaternion.

An IMU sensor data (e.g., acceleration) is a 3D vector $\vec{a}_{dev}$ in the device CF. During training, we use a random HACF at each epoch, which is defined by randomly rotating ground-truth trajectories on the horizontal plane. IMU data is transformed into the same HACF by the device orientation $R_{grv}$ and the same horizontal rotation $R_z$. The use of device orientations effectively incorporates sensor fusion\(^2\) into our data-driven system.

$$\vec{a}_{hacf} = R_z R_{grv} \vec{a}_{dev}. \quad (1)$$

At test time, we use the coordinate frame defined by system device orientations from Android, whose Z axis is aligned with gravity.

$$\vec{a}_{hacf} = R_{grv} \vec{a}_{dev}. \quad (2)$$

B. Backbone architectures

We present three RoNIN variants as shown in Fig. 3. We show that our key design principles consistently improve accuracy irrespective of the architecture in Sec. VI. In practice, one is free to choose any model, while the LSTM variant is more suitable when resources are limited, for example, mobile applications.

RoNIN ResNet: We take the 1D version of the standard ResNet-18 architecture [17] and add one fully connected layer with 512 units at the end to regress a 2D vector.

$$\vec{y}_i = f_{ResNet}(X_{i-200}) \quad (3)$$

At test time, we make predictions every five frames and integrate them to estimate motion trajectories.

RoNIN LSTM: We use a stacked unidirectional LSTM [18] while enriching its input feature by concatenating the output of a bilinear layer [20]. RoNIN-LSTM has three layers each with 100 units and regresses a 2D vector for each frame.

$$\vec{y}_{i-k:i} = f_{LSTM}(X_{i-k:i}) \quad (4)$$

RoNIN TCN: TCN [19] approximates many-to-many recurrent architectures with dilated causal convolutions. RoNIN TCN has six residual blocks with 16, 32, 64, 128, 72, and 36 channels, respectively, where a convolutional kernel of size 3 leads to the receptive field of 253 frames.

$$\vec{y}_{i-k:i} = f_{TCN}(X_{i-k:i}) \quad (5)$$

\(^2\)We utilize Android’s game rotation vector as device orientations.
C. Robust velocity loss

For position estimation, we regress velocity as it is naturally bounded by human motion dynamics. However, defining the ground-truth velocity for each frame amounts to computing the derivative of low-frequency VI-SLAM poses at much higher frame rate. This makes the ground-truth velocity noisy and ill-suited as supervision. We propose two robust velocity losses that increase the signal-to-noise-ratio for better motion learning.

Latent velocity loss: The output of RoNIN LSTM/TCN, $\bar{y}_{i-k}$, is summed by an integration layer and compared against the ground-truth positional difference over the same frame-window ($k = 400 \& 253$ for LSTM & TCN resp.). $\bar{p}_i$ is the position at frame $i$ in the horizontal plane.

$$loss = L_2(\bar{p}_i - \bar{p}_{i-k} - \int_{i-k}^{i} \bar{y}dt)$$

(6)

This loss simply enforces the integral of per-frame vectors to match displacement, hence the term latent velocity loss.

Strided velocity loss: For RoNIN ResNet, the network learns to predict positional difference over a stride of $k = 200$ instead of instantaneous velocities.

$$loss = MSE(\bar{p}_i - \bar{p}_{i-k}, \ \bar{y}_i)$$

(7)

D. RoNIN body heading network

We define body heading direction as a 2D unit vector perpendicular to the subject’s upper body (horizontal projection of Y axis in Fig. 4c). The heading direction becomes different from subject’s velocity direction in challenging motion cases (e.g. walking sideways) and becomes inherently ambiguous when a subject is stationary. Suppose one is sitting in a chair for 30 seconds. We need the IMU sensor data 30 seconds back in time to estimate the body heading, as IMU data have almost zero information after the sitting event.

We borrow the RoNIN LSTM architecture, which is capable of keeping a long memory, without the integration layer, and let the network predict a 2D unit vector $(a, b)$ pointing towards the body heading angle.

During training, we calculate loss against ground-truth body heading angles $\theta$ and add a normalization loss $\lambda = 1 - a^2_j - b^2_j$ to guide the network to predict valid trigonometric values. The angle is not regressed directly as it is ambiguous. We unroll the network over 1,000 steps for back-propagation. To avoid ambiguity when the subject is stationary, we use velocity magnitude $\bar{v}$ to identify initial stationary points, if any.

$$loss = \sum_{j=i-k}^{i} MSE((\sin \theta_j, \cos \theta_j), \ (a_j, b_j)) + \lambda$$

(8)

where $k = \arg\min_{j \in [i-1000, i]} |\bar{v}_j| > 0.1 m/s^{-1}$

V. EVALUATIONS: PRELIMINARIES

We implement the proposed architectures using PyTorch [21] and run our experiments using NVIDIA 1080Ti with 12GB GPU memory.

For RoNIN ResNet, we extract one training/validation sample every 10 frames. For RoNIN LSTM, we unroll the sequence to 400 steps once per $n$ frames, where $n$ is a random number between 50 and 150. Unrolled sequences are randomly batched to update network parameters. For RoNIN TCN, we construct one sample with 400 frames per $n$ frames, where $n$ is again a random number between 50 and 150.

For RoNIN ResNet (resp. RoNIN LSTM/TCN), we use a batch size of 128 (resp. 72), an initial learning rate of 0.0001 (resp. 0.0003), and ADAM optimizer while reducing the learning rate by a factor of 0.1 (resp. 0.75) if the validation loss does not decrease in 10 epochs, where the training typically converges after 100 (resp. 300/200) totalling 10 hours (resp. 40/30 hours). For linear layers we apply dropout with the probability 0.5 (resp. 0.2).

A. Competing methods

We conduct qualitative and quantitative evaluations of proposed algorithms on three datasets (RIDI, OXIOD, and RoNIN datasets) with four competing methods:

Naive double integration (NDI): We transform linear accelerations into global CF and integrate twice to get positions.

Pedestrian Dead Reckoning (PDR): We utilize a step-counting algorithm [22] to detect foot-steps and move the position along the device heading direction by a predefined distance of 0.67m per step.

Robust IMU Double Integration (RIDI): We use the official implementation [9]. For RIDI and OXIOD datasets, we train a separate model for each phone placement type. For RoNIN dataset, we train one unified model with 10% of RoNIN training data, since their Support Vector Regression model does not scale to larger dataset.

IONet: We use our local implementation, as the code is not publicly available. As in RIDI method, we train a unified model on RoNIN dataset, and a separate model for each placement type for RIDI and OXIOD datasets.

B. Device orientation handling

NDI, PDR, RIDI and RoNIN rely on external device orientation information. For fairness we use the device orientation estimated from IMU for testing\(^3\). During training, we use the same orientation for RIDI dataset. For OXIOD, 1) orientations from IMU are severely corrupted\(^4\), and 2) device CF from Vicon and Android device CF are not aligned identically or consistently. We use the ground-truth orientations from Vicon during training. For RoNIN, we use the estimated device orientations if the end-sequence alignment error is below 20\(^\circ\), otherwise choose the ground-truth to minimize erroneous samples during training.

C. Ground-truth alignment

RoNIN estimates trajectories in the global CF and we directly compare against the ground-truth for evaluations. For other methods we use ICP to align the first 5 seconds of the estimated and ground-truth trajectories before evaluation.

\(^3\)The orientation from Android APL called Game Rotation Vector

\(^4\)We believe that this is due to the poor bias calibration and the inappropriate choice of APIs using the magnetic field, which is usually distorted in indoor.
Table II
Position evaluation. We compare five competing methods: Naive Double Integration (NDI), Pedestrian Dead Reckoning (PDR), RIDI, IONet, and RoNIN (3 variants) on three datasets: RIDI dataset, OXIOD dataset and our new dataset. The top three results are highlighted in red, green, and blue colors per row.

| Test subjects | Metric | NDI | PDR | RIDI | IONet | RoNIN |
|---------------|--------|-----|-----|------|-------|-------|
|               |        | ATE | RTE | ATE  | RTE  | ATE  | RTE  | ATE  | RTE  | ATE  | RTE  | ATE  | RTE  |
| RIDI Dataset  | Seen   |     |     |      |      |      |      |      |      |      |      |      |      |
|               |        | 31.06 | 3.52 | 1.88 | 11.46 | 1.63 | 2.00 | 1.66 |      |      |      |      |      |
|               | Unseen |     |     |      |      |      |      |      |      |      |      |      |      |
|               |        | 32.01 | 1.98 | 1.71 | 12.50 | 1.89 | 2.08 | 1.88 |      |      |      |      |      |
| OXIOD Dataset | Seen   |     |     |      |      |      |      |      |      |      |      |      |      |
|               |        | 716.31 | 4.12 | 1.79 | 1.90 | 2.40 | 2.02 | 2.38 |      |      |      |      |      |
|               | Unseen |     |     |      |      |      |      |      |      |      |      |      |      |
|               |        | 1981.41 | 4.05 | 2.65 | 6.71 | 7.12 | 7.70 |      |      |      |      |      |      |
| RoNIN Dataset | Seen   |     |     |      |      |      |      |      |      |      |      |      |      |
|               |        | 675.21 | 29.54 | 17.06 | 31.07 | 3.54 | 4.18 | 4.38 |      |      |      |      |      |
|               | Unseen |     |     |      |      |      |      |      |      |      |      |      |      |
|               |        | 458.06 | 27.67 | 15.66 | 32.03 | 5.14 | 5.52 | 5.70 |      |      |      |      |      |
|               |        | 117.06 | 25.17 | 18.91 | 26.93 | 3.37 | 3.58 | 4.07 |      |      |      |      |      |

Fig. 5. Selected visualizations. We select 2 examples from each dataset and visualize reconstructed trajectories from competing methods. For each sequence we mark the trajectory length and report ATE and RTE of RoNIN method. The physical dimensions marked demonstrate that our method estimate trajectories with accurate scales. Examples from RoNIN dataset (left column) contains complex natural motions, where all other methods fail. RIDI dataset (middle) contains hard motions, such as extensive backward motion in the first example. OXIOD dataset (right) mostly contains short sequences with easy motions. However, our method gives large error for a few sequences (e.g. the bottom one) due to the large error in the provided device orientations.

Fig. 6. Selected visualization of heading angle estimations.
Fig. 7. The ratio of RoNIN testing sequences under different thresholds on the two metrics. For instance, the left graph shows the ratio of sequences where the ATE is below a certain threshold.

TABLE III
EVALUATION OF BODY HEADING ESTIMATION

| Test Subjects | Baseline | RoNIN Heading |
|---------------|----------|---------------|
|                | Seen     | Unseen        | Seen | Unseen |
| MSE (deg)     | 1.58     | 0.99          | 0.08 | 0.10   |
| MAE (degree)  | 90.60    | 89.10         | 14.40| 16.19  |

VI. EVALUATIONS

We conduct comprehensive evaluations on two tasks: 1) position estimation among five competing methods on three datasets; and 2) body heading estimation by our method on the RoNIN dataset.

A. Position evaluations

We use two standard metrics proposed in [23].

- Absolute Trajectory Error (ATE), defined as the Root Mean Squared Error (RMSE) between estimated and ground truth trajectories as a whole.
- Relative Trajectory Error (RTE), defined as the average RMSE over a fixed time interval, 1 minute in our evaluations. For sequences shorter than 1 minute, we compute the positional error at the last frame and scale proportionally.

Table II is our main result. Seen testing sets contains trajectories from subjects that are in the training sets and unseen for unseen subjects. Fig 5 shows selected visualizations of the reconstructed trajectories against the ground-truth. We show RoNIN ResNet from our methods.

RoNIN outperforms competing approaches on RIDI and RoNIN datasets with significant margins (Fig. 7). Most notably, no previous methods can handle natural complex motions presented in RoNIN dataset. All methods fail badly for a few sequences, where motions are not represented well in our training set (e.g. a phone in wildly moving handbag).

Both RIDI and RoNIN struggle on the OXIOD dataset despite their easy motions, where even PDR works well. This is simply due to their erroneous device orientation estimations, which RIDI and RoNIN rely on and assume to be correct. We expect their performance to improve with better bias calibration and the use of compass-free device orientation APIs.

B. Body heading evaluation

We use two metrics for evaluations: 1) Mean Squared Error (MSE) of the unit vector representation of the heading; and 2) Mean Angle Error (MAE) of the estimated heading in degrees. We compare against a simple baseline that reports the heading angles from the device orientations (i.e., device z-axis). We can evaluate only the heading difference in this baseline, and hence align the device heading and the ground-truth body heading by first 5 seconds of the sequence.

Table III and Fig. 6 show the results. We notice that our errors become significantly larger (up to 25°) for a few complex motion cases but are generally less than 15°. The baseline fails because it does not account for the orientation difference between the device and the body.

We observe that learning trajectory position and heading direction jointly performs poorly as 1) the history of IMU data needed is different for the two task and 2) balancing the loss function for the two tasks is non-trivial.

C. Ablation study

Fig. 8 shows the ablation study on the RoNIN dataset, demonstrating the effectiveness of the coordinate frame normalization and the robust velocity loss, by toggling these two features for the three architectures. We use raw IMU data in device CF and ground-truth instantaneous velocity for supervision as the baselines for coordinate frame normalization and robust velocity loss respectively.

We conclude that the coordinate frame normalization and the robust velocity losses improve ATE and RTE overall, while the former seems to have larger impact. In particular, ATE and RTE shows the lowest errors when both features are combined.

VII. DISCUSSIONS

This paper sets a new foundation for data-driven inertial navigation research by 1) the new benchmark with large and diverse quantity of IMU-motion data as in real day-to-day activities; 2) new neural inertial navigation architectures making significant improvements over challenging motion cases; and 3) qualitative and quantitative evaluations of the current competing methods over the three inertial navigation datasets. Two novel inertial learning tasks and models not restricted to few carrying types distinguish RoNIN from existing methods.

The major limitation of our approach comes from the reliance on the device orientation estimations. The performance degrades significantly given data with poor device orientations, which is the main focus of our future work.

Please refer to our website (http://ronin.cs.sfu.ca) for details on data pipeline, complete ablation study, more visualizations and discussion. We share our code and data to promote further research towards an ultimate anytime anywhere navigation system for everyone’s smartphone.
REFERENCES

[1] R. Mur-Artal and J. D. Tardós, “Visual-inertial monocular slam with map reuse,” IEEE Robotics and Automation Letters, vol. 2, no. 2, pp. 796–803, 2017.

[2] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, “Keyframe-based visual–inertial odometry using nonlinear optimization,” The International Journal of Robotics Research, vol. 34, no. 3, pp. 314–334, 2015.

[3] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, “On-manifold preintegration for real-time visual–inertial odometry,” IEEE Transactions on Robotics, vol. 33, no. 1, pp. 1–21, 2017.

[4] Apple, “Apple arkit,” https://developer.apple.com/arkit/.

[5] Google, “Arcore,” https://developers.google.com/ar/.

[6] Microsoft, “Hololens,” https://www.microsoft.com/microsoft-hololens/en-us.

[7] A. R. Jimenez, F. Seco, C. Prieto, and J. Guevara, “A comparison of pedestrian dead-reckoning algorithms using a low-cost mems imu,” in 2009 IEEE International Symposium on Intelligent Signal Processing. IEEE, 2009, pp. 37–42.

[8] A. Brajdic and R. Harle, “Walk detection and step counting on unconstrained smartphones,” in Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing. ACM, 2013, pp. 225–234.

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

[10] C. Chen, P. Zhao, C. X. Lu, W. Wang, A. Markham, and N. Trigoni, “Oxiod: The dataset for deep inertial odometry,” arXiv:1809.07491[cs.RO], 2018.

[11] R. E. Kalman, “A new approach to linear filtering and prediction problems,” Journal of basic Engineering, vol. 82, no. 1, pp. 35–45, 1960.

[12] J. Janardhanan, G. Dutta, and V. Tripuraneni, “Attitude estimation for pedestrian navigation using low cost mems accelerometer in mobile applications, and processing methods, apparatus and systems,” Apr. 8 2014, us Patent 8,694,251.

[13] M. Kourogi and T. Kurata, “A method of pedestrian dead reckoning for smartphones using frequency domain analysis on patterns of acceleration and angular velocity,” in Position, Location and Navigation Symposium-PLANS 2014, 2014 IEEE/ION. IEEE, 2014, pp. 164–168.

[14] C. Chen, X. Lu, A. Markham, and N. Trigoni, “Ionet: Learning to cure the curse of drift in inertial odometry,” in Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[15] A. M. Sabatini, “Quaternion-based extended kalman filter for determining orientation by inertial and magnetic sensing,” IEEE Transactions on Biomedical Engineering, vol. 53, no. 7, pp. 1346–1356, 2006.

[16] S. Madgwick, “An efficient orientation filter for inertial and inertial/magnetic sensor arrays,” Report x-io and University of Bristol (UK), vol. 53, no. 7, pp. 1346–1356, 2006.

[17] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[18] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] S. Bai, J. Z. Kolter, and V. Koltun, “An empirical evaluation of generic convolutional and recurrent networks for sequence modeling,” arXiv preprint arXiv:1803.01271, 2018.

[20] J. B. Tenenbaum and W. T. Freeman, “Separating style and content with bilinear models,” Neural computation, vol. 12, no. 6, pp. 1247–1283, 2000.

[21] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, “Automatic differentiation in PyTorch,” in MPS Autodiff Workshop, 2017.

[22] Q. Tian, Z. Salcic, I. Kevin, K. Wang, and Y. Pan, “An enhanced pedestrian dead reckoning approach for pedestrian tracking using smartphones,” in Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), 2015 IEEE Tenth International Conference on. IEEE, 2015, pp. 1–6.

[23] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of rgb-d slam systems,” in 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2012, pp. 573–580.