RIDI: Robust IMU Double Integration

Hang Yan[1(✉)], Qi Shan[2], and Yasutaka Furukawa[3]

[1] Washington University in St. Louis, St. Louis, USA
yanhang@wustl.edu
[2] Zillow Group, Seattle, USA
qis@zillowgroup.com
[3] Simon Fraser University, Burnaby, Canada
furukawa@sfu.ca

Abstract. This paper proposes a novel data-driven approach for inertial navigation, which learns to estimate trajectories of natural human motions just from an inertial measurement unit (IMU) in every smartphone. The key observation is that human motions are repetitive and consist of a few major modes (e.g., standing, walking, or turning). Our algorithm regresses a velocity vector from the history of linear accelerations and angular velocities, then corrects low-frequency bias in the linear accelerations, which are integrated twice to estimate positions. We have acquired training data with ground truth motion trajectories across multiple human subjects and multiple phone placements (e.g., in a bag or a hand). The qualitatively and quantitatively evaluations have demonstrated that our simple algorithm outperforms existing heuristic-based approaches and is even comparable to full Visual Inertial navigation to our surprise. As far as we know, this paper is the first to introduce supervised training for inertial navigation, potentially opening up a new line of research in the domain of data-driven inertial navigation. We will publicly share our code and data to facilitate further research (Project website: https://yanhangpublic.github.io/ridi).

1 Introduction

Accurate position estimation from an Inertial Measurement Unit (IMU) has long been a dream in academia and industry. IMU double integration is an approach with a simple principle: given a device rotation (e.g., from IMU), one measures an acceleration, subtracts the gravity, integrates the residual acceleration once to get velocities, and integrates once more to get positions. Dead-reckoning or step counting is another approach, which detects foot-steps to estimate the distance of travel and utilizes device rotations to estimate motion directions. IMU is in every smart-phone, is very energy-efficient (i.e., capable of running 24 h a day), and works anywhere even inside a bag or a pocket. A robust inertial navigation would be an ultimate anytime anywhere navigation system.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-01261-8_38) contains supplementary material, which is available to authorized users.

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11217, pp. 641–656, 2018.
https://doi.org/10.1007/978-3-030-01261-8_38

Unfortunately, the current state-of-the-art suffers from severe limitations. First, IMU double integration does not work unless one uses a million dollar military-grade IMU unit in a submarine, because small sensor errors and biases explode quickly in the double integration process. Second, dead-reckoning typically assumes that device rotations are exactly aligned with motion directions. This assumption almost always breaks in daily activities, as we move around devices from a hand to a pocket to a bag.

This paper proposes a simple data-driven approach that learns to estimate natural human motions only from IMU. Our key idea is that human motions are repetitive and consist of a small number of major modes. Our algorithm, dubbed Robust IMU Double Integration (RIDI), learns to regress walking velocities from IMU signals, while compensating for arbitrary device rotations with respect to the body. More precisely, RIDI regresses a velocity vector from the history of linear accelerations and angular velocities, then corrects low-frequency errors in the linear accelerations such that their integration matches the regressed velocities. A standard double integration is then used to estimate the trajectory from the corrected linear accelerations.

We have acquired IMU sensor data across 10 human subjects with 4 popular smartphone placements. The ground truth trajectories are obtained by a Visual Inertial Odometry system (i.e., a Google Tango phone, Lenovo Phab2 Pro) [12]. Our datasets consist of various motion trajectories over 150 min at 200 Hz. Our experiments have shown that RIDI produces motion trajectories comparable to the ground truth, with mean positional errors below 3%.

To our knowledge, this paper is the first to introduce supervised training for inertial navigation. Our algorithm is surprisingly simple, yet outperforms existing heuristic-based algorithms, and is even comparable to Visual Inertial Odometry. This paper could start a new line of research in data-driven Inertial Navigation. Commercial implications of the proposed research are also significant. IMUs are everywhere on the market, inside smartphones, tablets, or emerging wearable devices (e.g., Fitbit or Apple Watch). Almost everybody always carries one of these devices, for which RIDI could provide precise motion information with minimal additional energy consumption, a potential to enable novel location-aware services in broader domains. We will publicly share our code and data to facilitate further research.

2 Related Work

Motion tracking has long been a research focus in the Computer Vision and Robotics communities. Visual SLAM (V-SLAM) has made remarkable progress in the last decade [5,7,8,16,22,23], enabling a robust real-time system for indoors or outdoors up to a scale ambiguity. Visual-inertial SLAM (VI-SLAM) combines V-SLAM and IMU sensors, resolving the scale ambiguity and making the system further robust [13,18]. VI-SLAM has been used in many successful products such as Google Project Tango [12], Google ARCore [10], Apple ARKit [2] or Microsoft Hololens [21]. While being successful, the system suffers from two

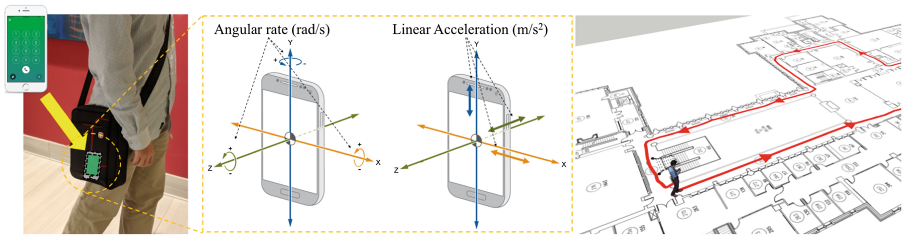


Fig. 1. Smartphones with motion sensors are ubiquitous in modern life. This paper proposes a novel data-driven approach for inertial navigation, which uses Inertial Measurement Unit (IMU) in every smartphone to estimate trajectories of natural human motions. IMU is energy-efficient and works anytime anywhere even for smartphones in your pockets or bags.

major drawbacks: (1) a camera must have a clear light-of-sight under well-lit environments all the time, and (2) the recording and processing of video data quickly drain a battery (Fig. 1).

IMU-only motion tracking has been successful in 3 DOF rotation estimation and been used in many recent Virtual Reality applications, such as Google Cardboard VR [11] or Samsung Gear VR [25]. While position estimation from IMU sensors has been a challenge due to sensor bias in the accelerometers, successful approaches exist for pedestrian motion tracking. The first family of these methods count foot-steps from accelerometers and multiply predefined step-lengths to estimate the translation [4,27]. Most these methods assume that the device orientation is aligned with the walking direction. Several methods seek to loosen this assumption by principal component analysis [15], forward-lateral modeling [6], or frequency domain analysis [17]. However, as shown in our experiments, these methods are based on heuristics and cannot handle more complex and varying motions in our database. Our approach is data-driven with supervised learning and makes significant improvements over these methods with a simple algorithm.

Another family of IMU-only pedestrian tracking methods use a foot-mounted IMU sensor and rely on the fact that the foot comes to a complete stop at every step [28]. The sensor drifts are eliminated through a zero-velocity constraint at each foot-step, producing impressive results. Our approach shares the idea of enforcing velocity constraints, but does not require a foot-mounted sensor.

WiFi signals are another information source for motion tracking without cameras in indoor environments [3,9,14,20]. A particle filter is applied on IMU, WiFi, and the map data to enable reliable motion tracking [19,24]. Inertial navigation is a critical component of WiFi based tracking. Our research is orthogonal and directly benefits these techniques.

3 Inertial Motion Trajectory Database

One contribution of the paper is a database of IMU sensor measurements and motion trajectories across multiple human subjects and multiple device placements. We have used a Google Tango phone, Lenovo Phab2 Pro, to record linear accelerations, angular velocities, gravity directions, device orientations (via Android APIs), and 3D camera poses. The camera poses come from the Visual Inertial Odometry system on Tango, which is accurate enough for our purpose (less than 1 m positional error after 200 m tracking). We make sure that the camera has a clear field-of-view all the time (See Fig. 2). This is only required when recording the ground truth trajectories for training and evaluation. Our method estimates motion trajectories solely from IMU data.

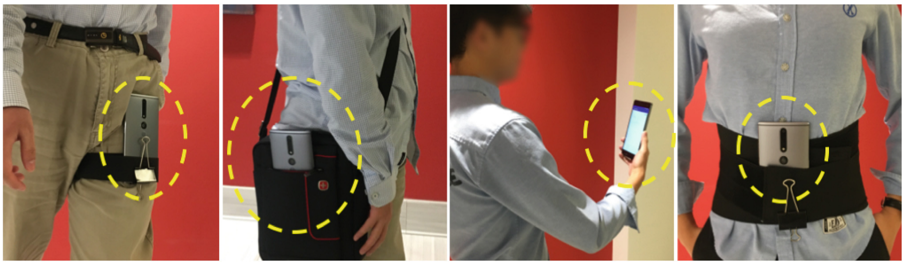


Fig. 2. We place a Tango phone in four popular configurations to collect training data. The ground truth motions come from Visual Inertial Odometry, and we have carefully designed the placements to make the camera always visible. From left to right: (1) in a leg pocket, (2) in a bag, (3) held by a hand, or (4) on a body (e.g., for officers).

We have collected more than 150 min of data at 200 Hz from ten human subjects under four popular smartphone placements with various motion types including walking forward/backward, side motion, or acceleration/deceleration. Asynchronous signals from various sources are synchronized into the time-stamps of Tango poses via linear interpolation. At total, our database consists of approximately two million samples.

4 Algorithm

The proposed algorithm, dubbed Robust IMU Double Integration (RIDI), consists of two steps. First, it regresses velocity vectors from angular velocities and linear accelerations (i.e., accelerometer readings minus gravity). Second, it estimates low-frequency corrections in the linear accelerations so that their integrated velocities match the regressed values. Corrected linear accelerations are double-integrated to estimate positions. We assume subjects walk on a flat floor. The regression and the position estimation are conducted on a 2D horizontal plane. We now explain a few coordinate frames and the details of the two steps.

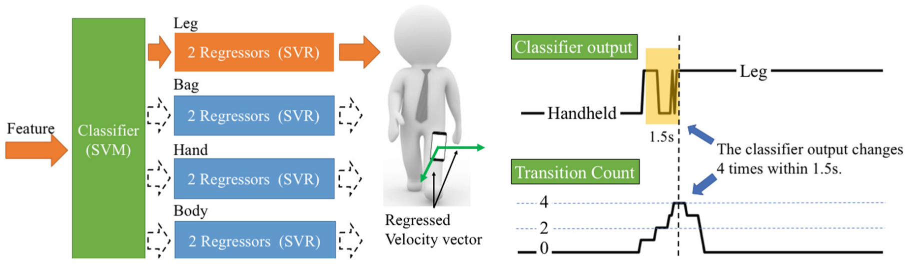
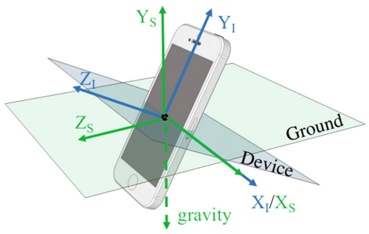


Fig. 3. Left: our cascaded regression model consists of one SVM and eight SVRs. SVM classifies the phone placement from the four types. Two type-specific SVRs predict a 2D velocity vector in the stabilized-IMU frame, ignoring the vertical direction. Right: we leverage the classifier output to handle transition periods. For the sample marked with the vertical dash line, the classifier output changes 4 times in the past 1.5 s, thus is identified to be within the transition period.

4.1 Coordinate Frames



The arbitrary device orientations with respect to the body makes it hard to infer the motion direction, being one of the core challenges of heuristic-based methods. Such arbitrariness also poses challenges to the regression task. With the assumption that the subject walks on a horizontal plane, we eliminate the device pitch and roll ambiguities by utilizing gravity estimations. We define a rectified coordinate frame, in which we train a velocity regressor to handle the remaining yaw ambiguity.

More precisely, we consider three coordinate frames in our algorithm. The first one is the world coordinate frame W, in which the output positions are estimated. W is set to be the global coordinate frame from the Android API at the first sample. The second one is the IMU/device coordinate frame I (marked with blue arrows in the right figure) in which IMU readings are provided by the Android APIs. Lastly, we leverage the gravity direction from the system to define our stabilized-IMU frame S, where the device pitch and roll are eliminated from I by aligning its y-axis with the gravity vector (see the green arrows in the right figure). This coordinate frame makes our regression task easier, since the regression becomes independent of the device pitching and rolling.

4.2 Learning to Regress Velocities

We learn to regress velocities in the stabilized IMU frame S. For each training sequence, we transform device poses (in W), and IMU readings (angular velocities and linear accelerations, in I) into S. The central difference generates velocity vectors from the transformed device poses (ignoring the vertical direction). To suppress high-frequency noise, we apply Gaussian smoothing with $\sigma = 2.0$ samples to 6 IMU channels, and with $\sigma = 30.0$ samples to 2 velocity

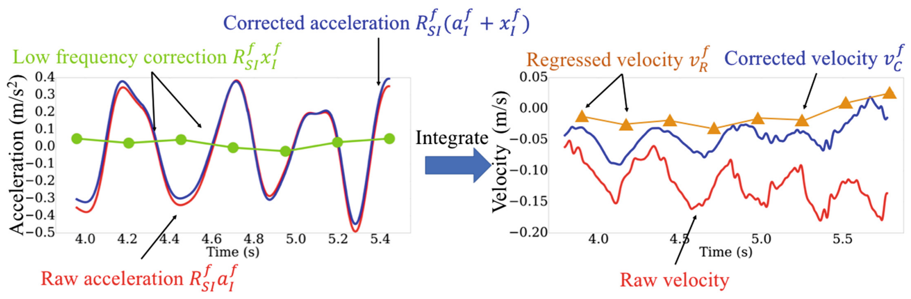


Fig. 4. Robust IMU double integration process. Our approach directly models the errors (green on the left) in the linear acceleration as a piecewise linear (thus low-frequency) function. We estimate parameters of this correction function so that the integration of the corrected linear accelerations (blue on the right) matches the regressed velocities (brown on the right). (Color figure online)

Table 1. Hyper-parameters for SVRs are found by the grid search: (1) C within a range of $[0.1, 100.0]$ with a multiplicative increment of 10; and (2) ϵ within a range of $[0.001, 1.0]$ with a multiplicative increment of 10.

	Leg	Bag	Hand	Body
C	1.0	10.0	10.0	1.0
ϵ	0.001	0.01	0.001	0.001

channels, respectively. We concatenate smoothed angular velocities and linear accelerations from the past 200 samples (i.e., 1 s) to construct a 1200 dimensional feature vector (Fig. 4).

People carry smartphones in different ways, exhibiting different IMU signal patterns. We assume that a phone is either (1) in a leg pocket, (2) in a bag, (3) held by a hand, or (4) on a body, and exploit this knowledge to propose a cascaded regression model (See Fig. 3). More precisely, a Support Vector Machine (SVM) first classifies the placement to be one of the above four types, then two type-specific ϵ-insensitive Support Vector Regression (SVR) [26] models estimate two velocity values independently (ignoring the vertical direction). The hyper-parameters for each model are tuned independently by the grid search and 3-fold cross validation, based on the mean squared error on the regressed velocities. The grid-search finds the soft-margin parameter $C = 10.0$ for SVM. Table 1 summarizes the chosen parameters for SVR models.

The above model assumes a phone being in the same placement all the time. To handle users switching a phone from one placement to another (e.g., picking up a phone from a leg pocket to a hand), we use a simple heuristic to identify *transition* periods, during which we specify the target velocity to be 0 without regression. Our observation is that the classifier makes near random predictions during transition periods. We inspect 10 contiguous classifier outputs in the past

1.5 s (i.e., one classification per 0.15 s) and declare the current sample to be in a transition if the classification results changed more than twice in the period (See Fig. 3 right).

4.3 Correcting Acceleration Errors

Predicted velocities provide effective cues in removing sensor noises and biases.[1] The errors come from various sources (e.g., IMU readings, system gravities, or system rotations) and interact in a complex way. We make a simplified assumption and model all the errors as a low-frequency bias in the linear acceleration. This approach is not physically grounded, but bypasses explicit noise/bias modeling and turns our problem into simple linear least squares.

We model the bias in the linear acceleration in the IMU/device coordinate frame I. To enforce the low-frequency characteristics, we represent the bias as linear interpolation of correction terms x_I^f at sub-sampled samples ($\mathcal{F}_1$), in particular, one term every 50 samples [29]. With abuse of notation, we also use x_I^f to denote interpolated acceleration correction (e.g., $x_I^{11} = 0.8x_I^1 + 0.2x_I^{51}$).

Our goal is to estimate $\{x_I^f\}$ at $\mathcal{F}_1$ by minimizing the discrepancy between the corrected velocities (v_C^f) and the regressed velocities (v_R^f) at sub-sampled samples $\mathcal{F}_2$ (once every 50 samples, to avoid evaluating SVRs at every sample for efficiency). The discrepancy is measured in the stabilized IMU frame S.

$$\min_{\{x_I^1, x_I^{51}, \cdots\}} \sum_{f \in \mathcal{F}_2} \left\| v_C^f - v_R^f \right\|^2 + \lambda \sum_{f \in \mathcal{F}_1} \left\| x_I^f \right\|^2,$$
$$v_C^f = \mathcal{R}_{SW}^f \sum_{f'=1}^{f} \mathcal{R}_{WI}^{f'} \left(a_I^{f'} + x_I^{f'} \right) \Delta t. \tag{1}$$

a_I^f denotes the raw linear acceleration in I. $\mathcal{R}_{AB}$ denotes the rotation that transforms a vector from coordinate frame $\mathcal{B}$ to $\mathcal{A}$. $\mathcal{R}_{WI}$ is the IMU orientation provided by the Android API. Suppose $\mathcal{R}_{SI}$ is the rotation that aligns the gravity vector to (0, 1, 0), $\mathcal{R}_{SW}$ can be then computed by left-multiplying $\mathcal{R}_{SI}$ to $\mathcal{R}_{IW}$. Δt is the time interval between samples, which is roughly 0.005 s under 200 Hz sampling rate.

The first term minimizes the discrepancy between the regressed (v_R^f) and the corrected (v_C^f) velocities. The corrected velocity (V_C^f) in the stabilized coordinate frame S is computed by (1) transforming each corrected linear acceleration into frame W by $\mathcal{R}_{WI}$; (2) integrating them in W; and (3) transforming to S by $\mathcal{R}_{SW}$.[2] Note that our regressor estimates the horizontal velocity, namely only the two entries in v_R^f without the vertical direction. We assume that subjects walk on the flat surface, and hence, fix the vertical component of v_R^f to be 0.

[1] Direct integration of the predicted velocities would produce positions but perform worse (See Sect. 6 for comparisons).

[2] We assume zero-velocity at the first sample, which is the case for our datasets. Relaxing this assumption is our future work.

The second term enforces l_2 regularization on the correction terms, which allows us to balance the velocity regression and the raw IMU signals. When λ is 0, the system simply integrates the regressed velocities without using raw IMU data. When λ is infinity, the system ignores the regressed velocities and performs the naive IMU double integration. We have used $\lambda = 0.1$ in our experiments. Double integration of the corrected accelerations produces our position estimations.

5 Implementation Details

We have implemented the proposed system in C++ with third party libraries including OpenCV, Eigen and Ceres Solver [1]. Note that our optimization problem (1) has a closed form solution, but we use Ceres for the ease of implementation. We have used a desktop PC with a Intel I7-4790 CPU and 32 GB RAM.

We have presented the algorithm as an offline batch method for clarity. It is fairly straightforward to implement an online algorithm, which has been used in all our experiments. Given sample i, the system returns the position at i by using corrected linear acceleration up to $i-1$. It also initializes the correction term for i as 0. Meanwhile, a second thread is launched once per 200 samples to solve for corrections within the last 1000 samples (with overlapping). In this way, the error is accumulated for no more than 1 s. The two expensive steps are the velocity regression and the optimization, which takes 26 ms and 17 ms on average, respectively. Our system processes 10,000 samples within 10 s, effectively achieving 1,000 fps on a desktop PC.

6 Experimental Results

We have acquired 74 motion sequences over 8 human subjects (marked as S1–S8), 4 different phone placements, and a variety of motion types. We have randomly selected 49 sequences for training and the remaining 25 sequences for testing. We have created one training/testing sample per 10 IMU samples, resulting in 109,365 training samples and 46,173 testing samples. We have also acquired 8 sequences from two unseen human subjects (S9, S10) and 4 sequences from an unseen device (Google Pixel XL) for testing.

6.1 Position Evaluations

Baseline Comparisons: Table 2 summarizes the quantitative evaluations on the accuracy of the final positions over 8 testing sequences (marked as T1–T8). We compared our method against 5 competing methods:

- **RAW** denotes the naive double integration with uncorrected linear accelerations (with system-level calibration and filtering).
- **STEP-ENH** denotes a recent step counting method [27]. The step length is set to the average of the ground truth step lengths over the whole training set.

- **STEP-FRQ** denotes another recent step counting method that uses frequency domain analysis to infer misalignment between the device orientation and the motion direction [17].[3] The step detection is provided by the Android API. The ground truth is used to set the step length as in STEP-ENH.
- **RIDI-MAG** is a variant of the proposed method. The regressed velocity vector consists of the magnitude and direction information. RIDI-MAG keeps the velocity magnitude, while replacing its direction by the system rotation through the Android API. RIDI-MAG cannot compensate for the device rotations with respect to the body.
- **RIDI-ORI** is another variant of RIDI that keeps the regressed velocity direction, while replacing the regressed velocity magnitude by the average of the ground truth values for each sequence.

Table 2. Positional accuracy evaluations. Each entry shows the mean positional error (in meters) and its percentage (inside parentheses) with respect to the trajectory length. The blue and the brown numbers show the best and the second best results.

Seq.	Place.	RAW	STEP-ENH	STEP-FRQ	RIDI-MAG	RIDI-ORI	RIDI
T1	Leg	15.43(23.41)	2.78(4.22)	6.64(10.08)	1.26(1.91)	1.93(2.93)	1.12(1.71)
T2	Leg	36.95(54.19)	3.91(5.74)	4.89(7.17)	1.03(1.52)	3.65(5.35)	1.00(1.47)
T3	Bag	55.35(35.73)	4.43(2.86)	10.67(6.89)	5.26(3.39)	9.74(6.29)	3.97(2.56)
T4	Bag	20.78(27.41)	2.10(2.76)	3.08(4.07)	1.32(1.74)	3.20(4.22)	1.14(1.51)
T5	Hand	172.8(112.2)	4.22(2.74)	14.98(9.73)	2.72(1.76)	10.36(6.73)	2.80(1.82)
T6	Hand	13.58(28.67)	4.38(9.25)	4.93(10.40)	4.88(10.30)	2.72(5.75)	1.22(2.57)
T7	Body	45.42(56.85)	15.17(18.98)	2.01(2.51)	10.78(13.49)	4.56(5.70)	1.71(2.14)
T8	Body	17.09(25.36)	0.94(1.40)	1.88(2.78%)	1.87(2.77)	2.66(3.94)	1.11(1.65)

For all the experiments, we align each motion trajectory to the ground truth by computing a 2D rigid transformation that minimizes the sum of squared distances for the first 10 s (2,000 samples). Table 2 shows that RIDI outperforms all the other baselines in most sequences, and achieves mean positional errors (MPE) less than 3.0% of the total travel distance, that is, a few meters after 150 m of walking. Figure 5 illustrates a few representative examples with regressed velocities. In T1, the phone is mounted over the leg pocket, where STEP-FRQ fails to infer correct motion direction due to abrupt leg movements. T5 is a case, in which the subject frequently changes the walking speeds. Both STEP-FRQ and RIDI-ORI fail for assuming a constant step frequency or velocity. In T7, the subject mixes different walking patterns, including backward motions. Only RIDI, STEP-FRQ and RIDI-ORI, which infer motion directions, perform well. Please visit our project website for more results and visualizations.

[3] Their algorithm has a heuristic to resolve the 180° ambiguity in the frequency analysis, but did not work well with our data. Our implementation favors this method by resolving the 180° ambiguity with the ground truth direction.

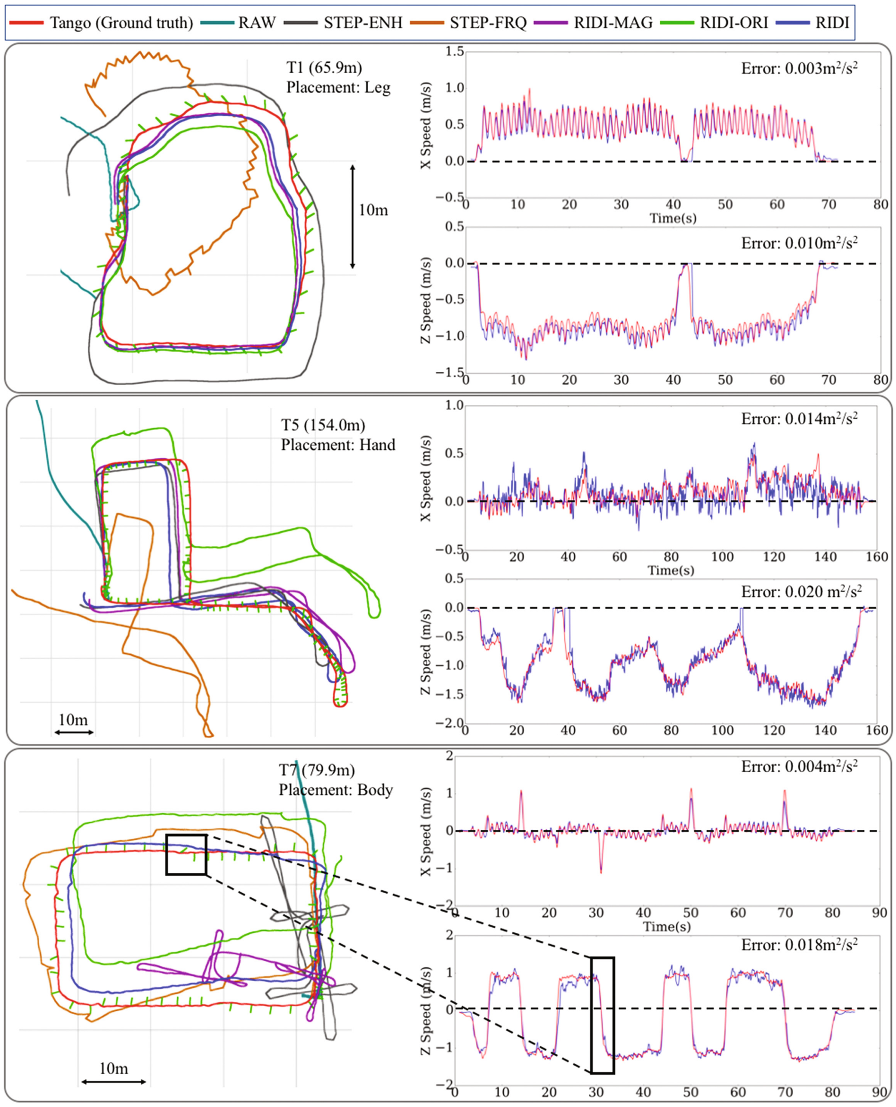


Fig. 5. Left: Motion trajectories from Tango, competing methods and RIDI. Short green segments indicate headings of device's X axis. Right: Regressed velocity vectors and their mean squared errors (MSE). In T1 (top row), the device is in a leg pocket and STEP-FRQ fails to infer correct orientation. In T5 (middle row), the subject frequently changes the speed, where STEP-FRQ and RIDI-ORI produce large errors for inaccurate motion frequencies and speed magnitudes. In T7 (bottom row), the subject mixes different walking patterns including 4 backward motions (the black rectangle is one place), where STEP-ENH and RIDI-MAG fails for not inferring velocity directions. Trajectories from the naive double integration(RAW) quickly diverge in all examples.

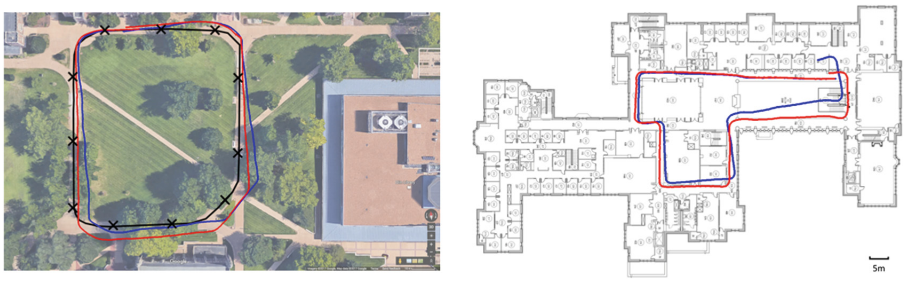


Fig. 6. Overlaying the trajectories with the online digital map (from Google Maps) or the floorplan image with the estimated scale. The red line marks the trajectory given by the Tango system and the blue line marks the trajectory given by our system. The accuracy of the Tango system degrades at outdoors in our experiments, so we manually drew the actual walking path with the black line at the left. (Color figure online)

Scale Consistency: One of the key advantages of the inertial or visual-inertial navigation is that the reconstruction is up to a metric-scale, which is not the case for image-only techniques such as visual-SLAM. Figure 6 shows that our trajectories are well aligned over a satellite or a floorplan image. We adjusted the scales (meters per pixel) based on the scale rulers, and manually specified the starting point and the initial orientation.

Parameter λ: Table 3 shows the impact of the parameter λ in Eq. 1, suggesting that it is important to integrate the velocity regression with the raw IMU acceleration data. Neither the regressed velocities (small λ) nor the naive double integration (large λ) performs well alone. We set $\lambda = 0.1$ in all experiments.

Table 3. The average MPE (as a ratio against the trajectory distance) over the testing sequences with different λ.

λ	0.0001	0.001	0.1	1.0	10,000
MPE	11.62%	1.49%	1.45%	1.47%	33.98%

Real-World Evaluation: we have qualitatively evaluated our system in a real world setting. A subject starts walking with the phone held by the hand. Along the route inside a large building, the subject performs several complex motions including putting the phone inside a bag, resting at a table, walking sideways and putting the phone into the leg pocket. Our method is able to estimate the trajectory under nature motions. The camera is blocked inside bag or pocket, therefore we omit the ground truth trajectory. See Fig. 7. Please visit our project website for detailed visualization.

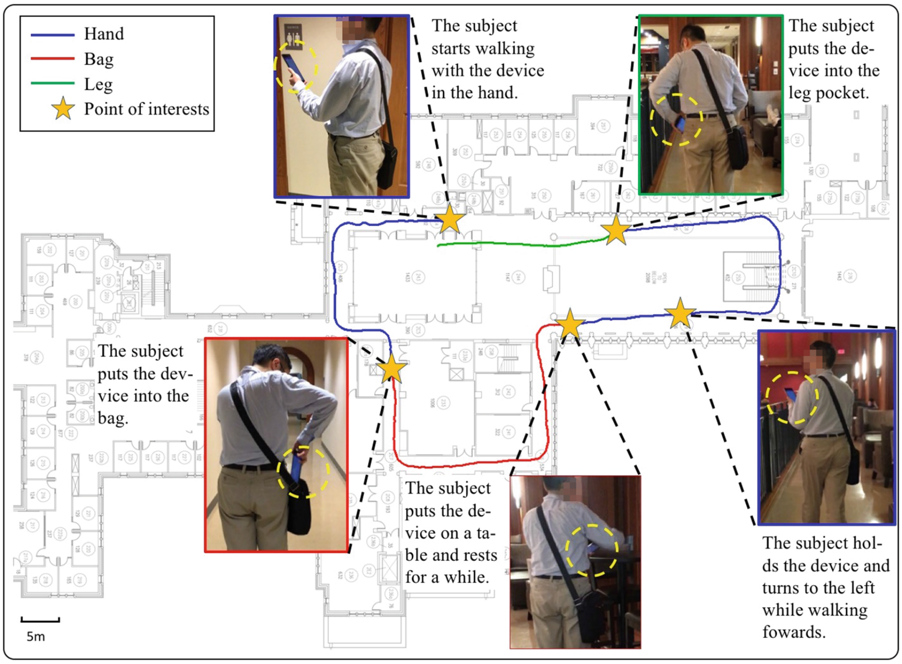

Hand
Bag
Leg
Point of interests
The subject starts walking with the device in the hand.
The subject puts the device into the leg pocket.
The subject puts the device into the bag.
The subject puts the device on a table and rests for a while.
The subject holds the device and turns to the left while walking fowards.
5m

Fig. 7. Real-world example with natural motions. The subject carries the phone with different placements (marked with colored lines) and performs several complex motions (marked with stars) along the route. Our method is able to estimate accurate trajectory only from IMU data. (Color figure online)

6.2 Velocity Evaluations

Our cascaded velocity regression achieves the mean squared errors of 0.017 [m^2/s^2] and 0.017 [m^2/s^2] on the X and Z axes on the testing set, respectively. We have also calculated the accuracy of the SVM classifier on the placement types, where the training and the testing accuracies are 97.00% and 96.22%, respectively. Lastly, we have evaluated the SVR regression model without the placement classification. The mean squared errors on the X and Z axes are 0.028 [m^2/s^2] and 0.032 [m^2/s^2], respectively, which are worse than our cascaded model. Acquiring more training data and evaluating the accuracy of more data-hungry methods such as deep neural networks is one of our future works.

6.3 Generalization

Unseen Devices: Considering the impact to commercial applications, the generalization capability to unseen devices is of great importance. We have used another device (Google Pixel XL) to acquire additional testing sequences. The subjects also carried the Tango phone to obtain ground truth trajectories. The sequence contains a quick rotation motion at the beginning to generate distinctive peaks in the gyroscope signals, which are used to synchronize data from the two devices. We register the estimated trajectories to the ground truth by the same process as before. Figure 8 shows that our system generalizes reasonably well under all placement types, in particular, still keeping the mean positional errors below 3%.

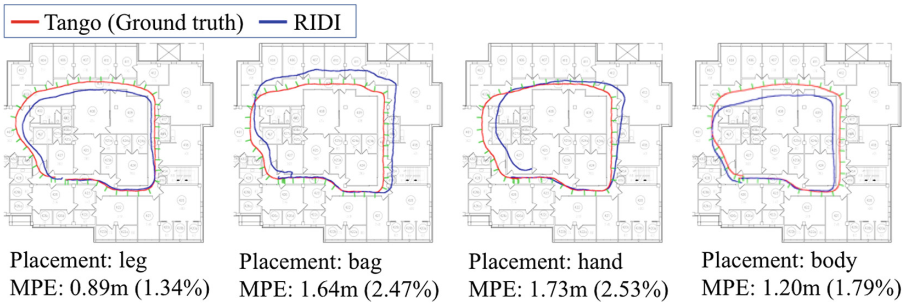


Fig. 8. Generalization to an unseen device (Google Pixel XL).

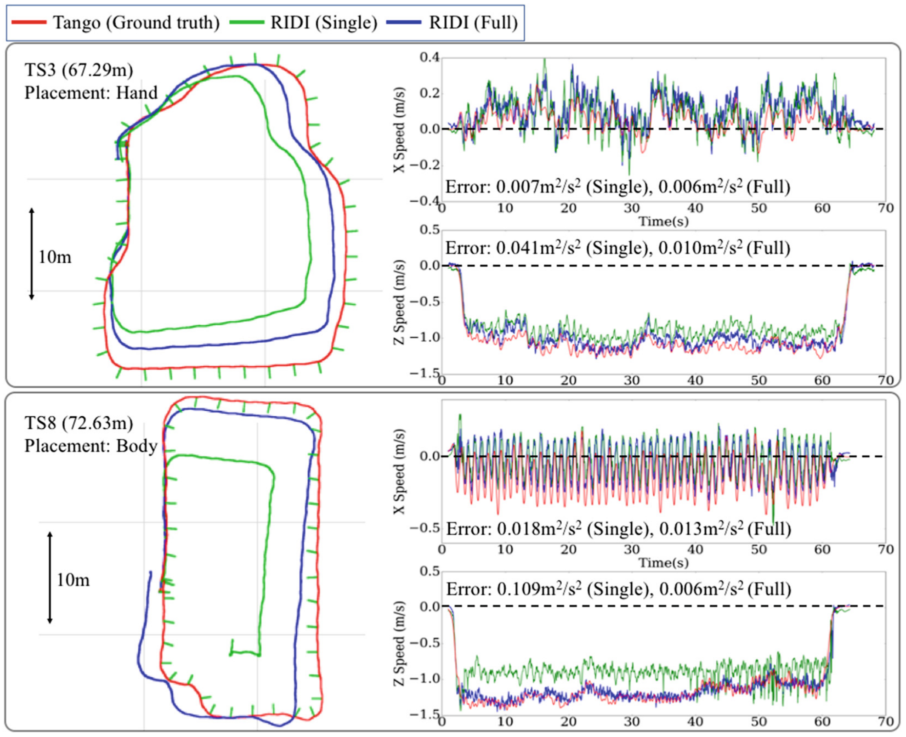


Fig. 9. Generalization to unseen subjects. We varied the number of human subjects in the training data and evaluated two RIDI models for unseen testing subjects. RIDI (single) uses training data only from 1 subject, while RIDI (Full) uses training data from the 8 subjects.

Unseen Subjects: The last experiment evaluates the generalization capability to unseen subjects (marked as S9 and S10). These two subjects have no prior knowledge of our project and we asked them to walk in their own ways. We have trained two RIDI models with different training sets. RIDI (Single) is trained on data only from 1 subject (S1). RIDI (Full) is trained on data from the 8 subjects (S1–S8). For fair comparisons, we have down-sampled the larger training set so that both sets contain around 28,000 training samples. Figure 9 and Table 4 demonstrate that the Full model generalizes well, in particular, below 4% MPE in most cases. However, the system performs worse in some sequences. Another important future work is to push the limit of the generalization capability by collecting more data and designing better regression machineries.

Table 4. Generalization to unseen subjects. The forth and fifth columns are the mean squared errors on the regressed velocities along the two horizontal axes. Last two columns are the mean positional errors (MPE) in meter and their percentage w.r.t. trajectory lengths (inside parentheses). The model trained on more subjects generalizes better.

Seq.	Subject	Place.	Reg. Err. (Single)	Reg. Err. (Full)	MPE (Single)	MPE (Full)
TS1	S9	Leg	(0.041, 0.072)	(0.023, 0.023)	3.51(4.83)	2.65(3.65)
TS2	S9	Bag	(0.108, 0.128)	(0.053, 0.009)	5.34(7.06)	2.88(3.80)
TS3	S9	Hand	(0.007, 0.041)	(0.006, 0.010)	2.35(3.49)	1.37(2.03)
TS4	S9	Body	(0.021, 0.053)	(0.011, 0.023)	8.43(12.42)	2.89(4.26)
TS5	S10	Leg	(0.031, 0.026)	(0.018, 0.018)	4.48(3.14)	3.22(2.26)
TS6	S10	Bag	(0.045, 0.024)	(0.013, 0.014)	2.71(1.94)	1.92(1.37)
TS7	S10	Hand	(0.021, 0.027)	(0.009, 0.008)	2.33(1.67)	1.25(0.89)
TS8	S10	Body	(0.018, 0.109)	(0.013, 0.006)	4.450(6.19)	1.35(1.86)

7 Conclusion

The paper proposes a novel data-driven approach for inertial navigation that robustly integrates linear accelerations to estimate motions. Our approach exploits patterns in natural human motions, learns to regress a velocity vector, then corrects linear accelerations via simple linear least squares, which are integrated twice to estimate positions. Our IMU-only navigation system is energy efficient and works anywhere even inside a bag or a pocket, yet achieving comparable accuracy to a full visual inertial navigation system to our surprise. Our future work is to collect a lot more training data across more human subjects on more devices, and learn a universal velocity regressor that works for anybody on any device. Another important future work is to deploy the system on computationally less powerful mobile devices. The impact of the paper to both scientific and industrial communities could be profound. This paper has a potential to open up a new line of learning based inertial navigation research. Robust anytime-anywhere navigation system could immediately benefit a wide range of industrial applications through location-aware services including online advertisements, digital mapping, navigation, and more.

Acknowledgement. This research is partially supported by National Science Foundation under grant IIS 1540012 and IIS 1618685, Google Faculty Research Award, and Zillow gift fund.

References

1. Agarwal, S., Mierle, K., et al.: Ceres solver. http://ceres-solver.org
2. Apple: Apple arkit. https://developer.apple.com/arkit/
3. Bahl, P., Padmanabhan, V.N.: RADAR: An in-building RF-based user location and tracking system. In: Proceedings of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies, INFOCOM 2000, vol. 2, pp. 775–784. IEEE (2000)
4. Brajdic, A., Harle, R.: Walk detection and step counting on unconstrained smartphones. In: Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing, pp. 225–234. ACM (2013)
5. Cadena, C., et al.: Past, present, and future of simultaneous localization and mapping: toward the robust-perception age. IEEE Trans. Robot. **32**(6), 1309–1332 (2016)
6. Chowdhary, M., Sharma, M., Kumar, A., Dayal, S., Jain, M.: Method and apparatus for determining walking direction for a pedestrian dead reckoning process. US Patent App. 13/682,684, 22 May 2014
7. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: Monoslam: real-time single camera slam. IEEE Trans. Pattern Anal. Mach. Intell. **29**(6), 1052–1067 (2007)
8. Engel, J., Koltun, V., Cremers, D.: Direct sparse odometry. IEEE Trans. Pattern Anal. Mach. Intell. **40**(3), 611–625 (2018)
9. Ferris, B., Fox, D., Lawrence, N.: WiFi-SLAM using Gaussian process latent variable models. In: Proceedings of IJCAI 2007, pp. 2480–2485 (2007)
10. Google: Arcore. https://developers.google.com/ar/
11. Google: Cardboard. https://vr.google.com/cardboard/
12. Google: Project tango. https://get.google.com/tango/
13. Hesch, J.A., Kottas, D.G., Bowman, S.L., Roumeliotis, S.I.: Camera-IMU-based localization: observability analysis and consistency improvement. Int. J. Robot. Res. **33**(1), 182–201 (2014)
14. Huang, J., Millman, D., Quigley, M., Stavens, D., Thrun, S., Aggarwal, A.: Efficient, generalized indoor wifi graphslam. In: IEEE International Conference on Robotics and Automation, pp. 1038–1043 (2011)
15. Janardhanan, J., Dutta, G., Tripuraneni, V.: Attitude estimation for pedestrian navigation using low cost mems accelerometer in mobile applications, and processing methods, apparatus and systems. US Patent 8,694,251, 8 April 2014
16. Klein, G., Murray, D.: Parallel tracking and mapping for small AR workspaces. In: ISMAR, pp. 225–234. IEEE (2007)
17. Kourogi, M., Kurata, T.: A method of pedestrian dead reckoning for smartphones using frequency domain analysis on patterns of acceleration and angular velocity. In: 2014 IEEE/ION Position, Location and Navigation Symposium-PLANS 2014, pp. 164–168. IEEE (2014)

18. Leutenegger, S., Lynen, S., Bosse, M., Siegwart, R., Furgale, P.: Keyframe-based visual–inertial odometry using nonlinear optimization. Int. J. Robot. Res. **34**(3), 314–334 (2015)
19. Li, F., Zhao, C., Ding, G., Gong, J., Liu, C., Zhao, F.: A reliable and accurate indoor localization method using phone inertial sensors. In: Proceedings of the 2012 ACM Conference on Ubiquitous Computing, pp. 421–430. ACM (2012)
20. Lim, C.H., Wan, Y., Ng, B.P., See, C.M.S.: A real-time indoor WiFi localization system utilizing smart antennas. IEEE Trans. Consum. Electron. **53**(2), 618–622 (2007)
21. Microsoft: Hololens. https://www.microsoft.com/microsoft-hololens/en-us
22. Mur-Artal, R., Montiel, J.M.M., Tardos, J.D.: ORB-SLAM: a versatile and accurate monocular slam system. IEEE Trans. Robot. **31**(5), 1147–1163 (2015)
23. Newcombe, R.A., Lovegrove, S.J., Davison, A.J.: DTAM: dense tracking and mapping in real-time. In: ICCV. pp. 2320–2327. IEEE (2011)
24. Racko, J., Brida, P., Perttula, A., Parviainen, J., Collin, J.: Pedestrian dead reckoning with particle filter for handheld smartphone. In: 2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN), pp. 1–7. IEEE (2016)
25. Samsung: Samsung gear VR. http://www.samsung.com/global/galaxy/gear-vr/
26. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Stat. Comput. **14**(3), 199–222 (2004)
27. Tian, Q., Salcic, Z., Kevin, I., Wang, K., Pan, Y.: An enhanced pedestrian dead reckoning approach for pedestrian tracking using smartphones. In: 2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP), pp. 1–6. IEEE (2015)
28. Yun, X., Bachmann, E.R., Moore, H., Calusdian, J.: Self-contained position tracking of human movement using small inertial/magnetic sensor modules. In: 2007 IEEE International Conference on Robotics and Automation, pp. 2526–2533. IEEE (2007)
29. Zhou, Q.Y., Koltun, V.: Simultaneous localization and calibration: self-calibration of consumer depth cameras. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 454–460 (2014)