RIDI: Robust IMU Double Integration

Hang Yan
Washington University in St. Louis
yanhang@wustl.edu

Qi Shan
Zillow Group
qis@zillow.com

Yasutaka Furukawa
Simon Fraser University
furukawa@sfu.ca

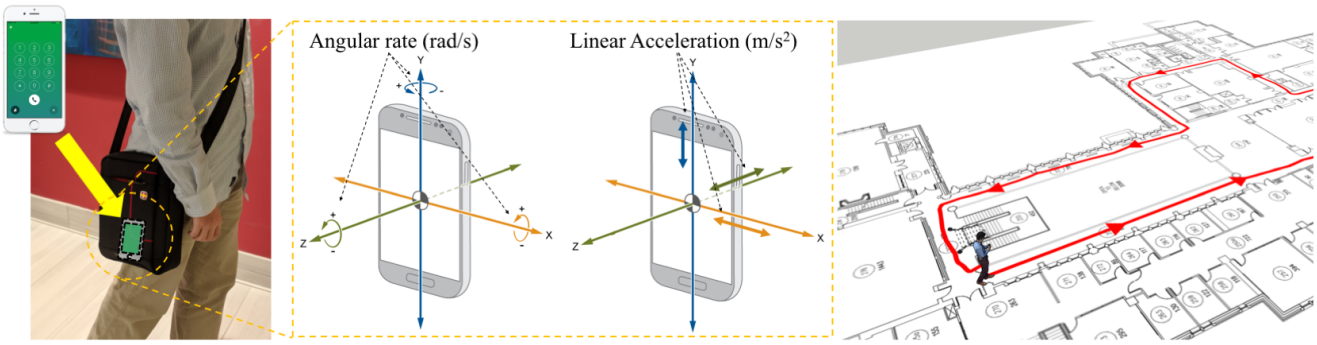


Figure 1: Smartphones with motion sensors are ubiquitous in modern life. This paper proposes a novel data-driven approach for inertial navigation, which uses Inertial Measurement Unit (IMU) in every smartphone to estimate trajectories of natural human motions. IMU is energy-efficient and works anytime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

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 motions 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 algorithm has surprisingly shown comparable results to full Visual Inertial navigation. To our knowledge, this paper is the first to integrate sophisticated machine learning techniques with 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. [1]

1. Introduction

IMU double integration for motion estimation has long been a dream for academic researchers and industry engineers. IMU is 1) energy-efficient, capable of running 24 hours a day without draining a battery; and 2) works anywhere even inside a bag or a pocket. The theory is simple: given the device orientation (e.g., via Kalman filter on IMU signals), one subtracts the gravity from the device acceleration, integrates the residual accelerations once to get velocities, and integrates once more to get positions. Unfortunately, small sensor errors or biases explode quickly in the double integration process. Such systems do not work in practice, unless one uses a million dollar military-grade IMU unit, often found on submarines.

Our key idea is that human motions are repetitive and consist of a small number of major modes. Pedometry precisely exploits this property to estimate the travel distance by step-counting. Together with standard IMU-based rotation estimation, it would appear relatively easy to estimate reasonable walking trajectories just from IMUs. However, this approach assumes that the device rotation is exactly aligned with the walking direction, which is not the case for our smartphones in our hands, bags, or leg pockets. The approach also fails for side motions or backward motions.

Our algorithm, dubbed Robust IMU Double Integration (RIDI), takes a data driven approach and learns to regress the instantaneous motion (i.e., a velocity) from IMU sig-

[1] Project website: https://yanhangpublic.github.io/ridi/index.html

nals, while automatically adjusting arbitrary device rotations with respect to a 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 to be compatible with the regressed velocities. A standard double integration is used to estimate the trajectory from the corrected linear accelerations.

We have acquired IMU sensor data across six human subjects with four popular smartphone placements. The ground-truth trajectories are obtained by a Visual Inertial Odometry system (i.e., a Google Tango phone, Lenovo Phab2 Pro) [4]. Our datasets consist of various motion trajectories over 100 minutes at 200Hz. 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 integrate sophisticated machine learning techniques with inertial navigation, and 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 out code and data to facilitate further research.

2. Related Work

Visual SLAM (V-SLAM) [10] has made remarkable progress in the last decade [16, 20, 21, 11], 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, resolving the scale ambiguity and making the system further robust. VI-SLAM is used in many successful products such as Google Project Tango [4], Google ARCore [3], Apple ARKit [1] or Microsoft Hololens [5]. While being successful, the system suffers from two 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.

IMU-based rotation estimation has been successful, used in many recent Virtual Reality applications such as Google Cardboard VR [13] or Samsung Gear VR [22]. IMU has also been proven effective for gait recognition [19], activity recognition [24], or step-counting [9, 15]. Step-counting, in particular, can lead to travel distance estimation, but the motion estimation is a challenge as the device orientation and the motion direction are not always aligned. In general, IMU-based position estimation still requires impractical assumptions, such as IMU on a foot [25] or presence of map data [17]. This paper integrates machine learning and inertial navigation to propose a robust data-driven approach without heuristics or impractical assumptions.

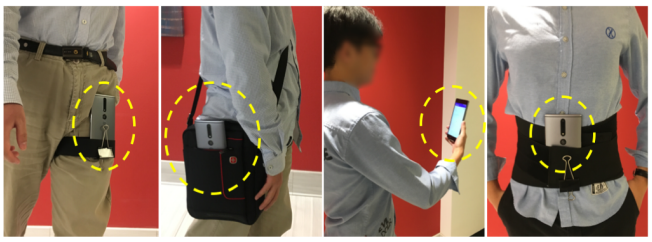


Figure 2: We place a Tango phone in four popular configurations to collect training data. The ground-truth motions come from Visual Inertial Odometry, and we have carefully designed the placements to make the camera always visible. From left to right: 1) in a leg pocket, 2) in a bag, 3) in a hand, or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

WiFi-based position tracking could also be a low-energy anytime-anywhere solution [18, 8, 12, 14]. However, these technologies are orthogonal to our work and we do not consider them as competing methods. State-of-the-art WiFi-based tracking system, WiFiSlam [14], critically depends on inertial navigation, and this paper directly benefits WiFi-based tracking systems.

3. Inertial 3D Motion Database

One contribution of the paper is a database of IMU sensor measurements and 3D 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, and we make sure that the camera has a clear field-of-view all the time (See Fig. 2).

We have collected more than 100 minutes of data at 200Hz from six 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 as a pre-processing step.

4. Algorithm

The proposed algorithm, dubbed Robust IMU Double Integration (RIDI), consists of two steps. First, RIDI regresses a velocity vector from angular velocities and linear accelerations (i.e., accelerometer readings minus gravity). Second, RIDI estimates low-frequency corrections in the linear accelerations so that their integrated velocities match the regressed values. Corrected linear accelerations are integrated to estimate positions. We assume that subjects walk

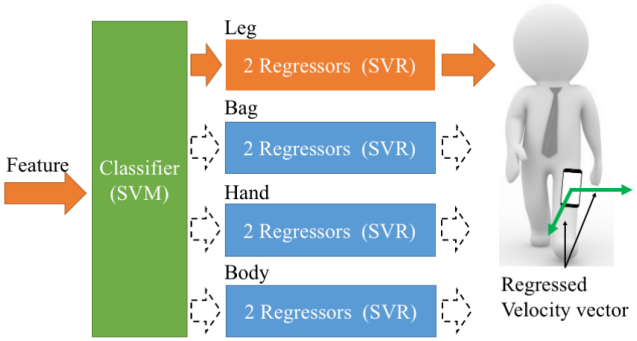
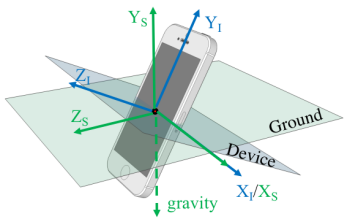


Figure 3: Our cascaded regression consists of one SVM and eight SVRs. SVM classifies the phone placement from the four types. Two type-specific SVRs predict a 2D velocity in the stabilized-IMU frame, ignoring the vertical direction.

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.

4.1. Coordinate frames



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 frame. 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 system 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 tilting 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 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 $\sigma = 2.0$ frames to 6 IMU channels, and with $\sigma = 30.0$ frames to 2 velocity channels, respectively. We concatenate smoothed angular velocities and linear accelerations from the past 200 frames (i.e., 1 second) to construct a 1200 dimensional feature vector.

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

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) hand-held, or 4) on 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) [23] models estimate two velocity values independently (ignoring the vertical direction). The hyper-parameters for each of the SVM/SVR models 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.

4.3. Correcting acceleration errors

Predicted velocities provide effective cues in removing sensor noises and biases.[2] 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 frames ($\mathcal{F}_1$), in particular, one term every 50 frames [26]. 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 frames $\mathcal{F}_2$ (once every 50 frames, to avoid evaluating SVRs at every frame for efficiency). The discrepancy is measured in the stabi-

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

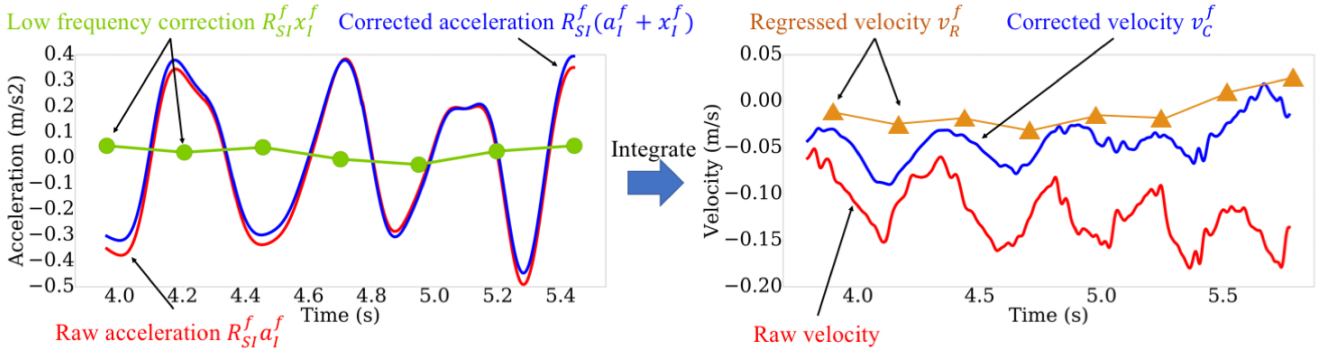


Figure 4: Robust IMU double integration process. Our approach directly models the errors (green on the left) in the linear acceleration as a piecewise linear (thus low-frequency) function. We estimate parameters of this correction function so that the integration of the corrected linear accelerations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

lized IMU frame S.

$$\min_{\{x_I^1, x_I^{51}, \cdots\}} \sum_{f \in \mathcal{F}_2} \left\| v_C^f - v_R^f \right\|^2 + \lambda \sum_{f \in \mathcal{F}_1} \left\| x_I^f \right\|^2, \qquad (1)$$
$$v_C^f = \mathcal{R}_{SW}^f \sum_{f'=1}^{f} \mathcal{R}_{WI}^{f'} \left(a_I^{f'} + x_I^{f'} \right).$$

a_I^f denotes the raw linear acceleration in I. $\mathcal{R}_{AB}$ denotes the rotation that transforms a vector in coordinate frame $\mathcal{B}$ to $\mathcal{A}$. The corrected velocity (v_C^f) is simply the integration of the corrected accelerations, except that the summation must occur in a consistent coordinate frame, for example W (but not S, which changes every frame). [3]

The first term minimizes the velocity discrepancy. Note that our regressor estimates the horizontal velocity, namely only the two entries in v_R^f except 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. The second term enforces l_2 regularization, 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

We have implemented the proposed system in C++ with third party libraries including OpenCV [6], Eigen [2], and Ceres Solver [7]. 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 equipped with a Intel I7-4790 CPU and 32GB RAM.

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

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. The online algorithm keeps track of two threads. The first thread updates correction terms once every 200 frames using the latest 1,000 frames. The second thread conducts double-integration every frame to produce a position estimation. The correction terms are initialized to 0 before being updated by the first thread. The two expensive steps are the SVR regression and the correction optimization, which takes 19ms and 15ms on the average, respectively. Our system processes 10,000 frames within 5 seconds, effectively achieving 2,000 fps on a desktop PC.

6. Experimental results

We have acquired 60 motion sequences over 4 human subjects (marked as S1-S4), 4 different phone placements, and a variety of motion types. We have randomly selected 40 sequences for training and the remaining 20 sequences for testing. We have created one data sample per 10 frames, resulting in 67,649 training samples and 44,236 testing samples. We have also acquired additional sequences from two unseen human subjects and one unseen device 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 4 competing methods:

- **RAW** denotes the naive double integration with the raw linear accelerations.
- **STEP** denotes the standard step-counting method. Step counts are provided by the Android API. We use the ground-truth motion to compute the length of each step, and use their average to determine the step-length to be used by this method for each sequence. The motion direction is set

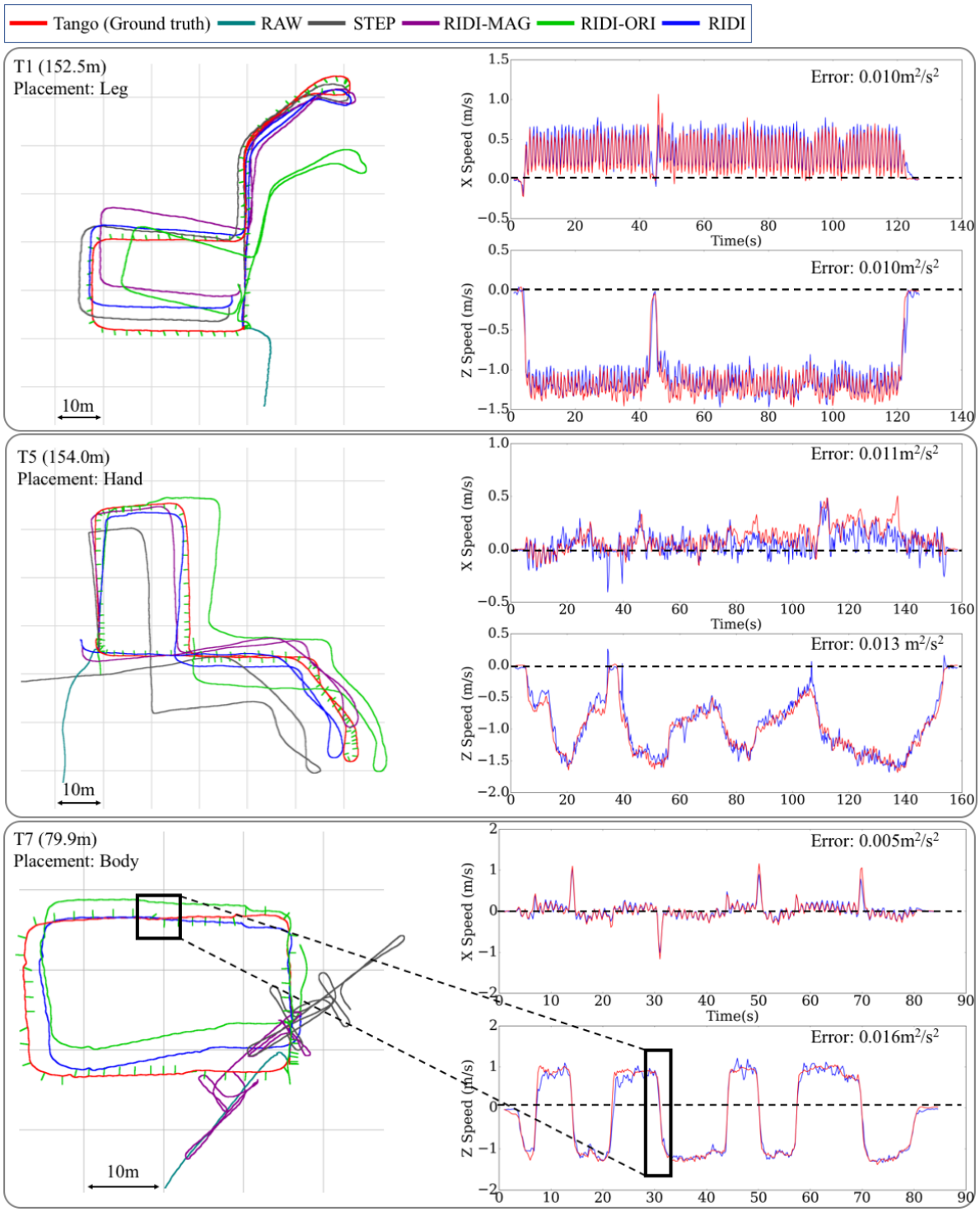


Figure 5: Left: Motion trajectories from Tango, competing methods, and RIDI. Right: Regressed velocity vectors and their mean squared errors (MSE). Short green segments indicate headings of device's X axis. The naive double integration (RAW) fails in many examples. In T5 (middle row), the subject frequently changes the speed, where STEP and RIDI-ORI produce large errors for inaccurate speed magnitudes. In T7 (bottom row), the subject mixes different walking patterns including 4 backward motions (the black rectangle is one place), where STEP and RIDI-MAG fails for not inferring velocity directions.

Table 2: Positional accuracy evaluations. Each entry shows the mean positional error and its ratio (inside a parentheses) with respect to the trajectory distance. The blue and the brown numbers show the best and the second best results.

Sequence	Placement	RAW [m]	STEP [m]	RIDI-MAG [m]	RIDI-ORI [m]	RIDI [m]
T1	Leg	55.92(36.95%)	1.64(1.08%)	2.37(1.57%)	6.22(4.11%)	1.50(0.99%)
T2	Leg	57.06(83.68%)	0.76(1.11%)	1.31(1.92%)	3.29(4.82%)	1.23(1.81%)
T3	Bag	71.93(46.43%)	6.70(4.33%)	4.44(2.87%)	6.40(4.13%)	4.07(2.63%)
T4	Bag	18.18(23.98%)	3.13(4.13%)	0.80(1.06%)	3.14(4.15%)	0.60(0.79%)
T5	Hand	180.03(116.91%)	9.32(6.05%)	2.54(1.65%)	8.37(5.44%)	0.91(0.59%)
T6	Hand	28.93(61.09%)	5.56(11.74%)	4.23(8.94%)	3.33(7.02%)	1.06(2.23%)
T7	Body	56.06(70.17%)	16.28(20.37%)	10.87(13.60%)	2.88(3.61%)	1.77(2.22%)
T8	Body	20.14(29.41%)	0.96(1.41%)	0.75(1.09%)	2.02(2.95%)	0.77(1.12%)

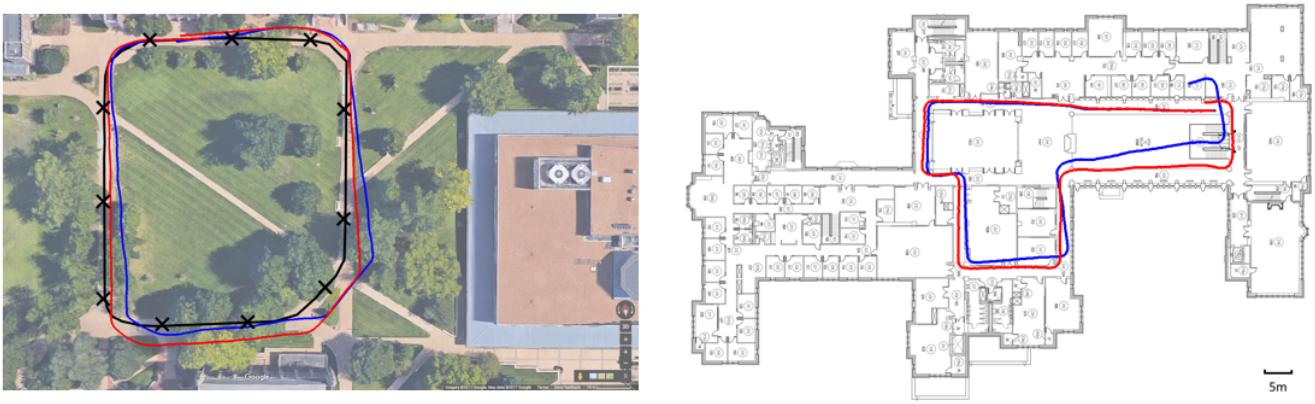


Figure 6: Overlaying the trajectories with the online digital map (from Google Maps) or the floorplan image with the estimated scale. The red line marks the trajectory given by the Tango system and the blue line marks the trajectory given by our system. The accuracy of the Tango system degrades at outdoors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

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%

to the device rotation given by the Android API. [4]

• **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 the same as RIDI-MAG except that it keeps the regressed direction, while replacing the regressed magnitude by the average of the ground-truth values for each sequence.

[4]We disable the magnetometer in our experiments, which damages rotation estimations due to instable magnetic fields.

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 seconds (2,000 frames). Table 2 illustrates 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 meters of walking. Figure 5 illustrates a few representative examples with regressed velocities. T5 is a case, in which the subject frequently changes the walking speeds. RIDI-ORI fails for assuming a constant speed and STEP fails for inaccurate stride lengths. T6 and T7 are cases, in which the subjects mix different walking patterns, including backward motions. Only RIDI and RIDI-ORI, which infer motion directions, perform well. Please refer to the supplementary material for more results and visualizations.

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

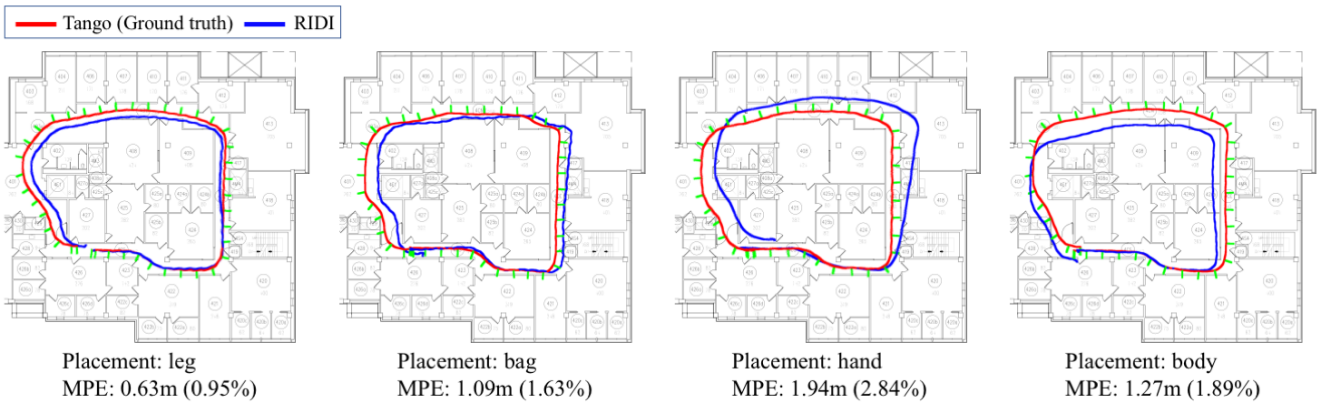


Figure 7: Generalization to an unseen device (Google Pixel XL).

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. The model trained on more subjects generalizes better.

Sequence	Subject	Placement	Reg. Error (Single)	Reg. Error (Full)	MPE (Single)	MPE (Full)
TS1	S5	Leg	(0.051, 0.318)	(0.027, 0.089)	5.98(8.25%)	3.51(4.83%)
TS2	S5	Bag	(0.108, 0.101)	(0.062, 0.060)	3.81(5.03%)	2.67(3.52%)
TS3	S5	Hand	(0.007, 0.025)	(0.005, 0.017)	1.50(2.22%)	1.27(1.89%)
TS4	S5	Body	(0.027, 0.034)	(0.018, 0.028)	2.20(3.24%)	2.16(3.18%)
TS5	S6	Leg	(0.056, 0.308)	(0.050, 0.110)	6.00(8.69%)	3.28(4.75%)
TS6	S6	Bag	(0.316, 0.427)	(0.195, 0.322)	11.79(11.36%)	8.36(8.05%)
TS7	S6	Hand	(0.049, 0.045)	(0.029, 0.029)	2.41(1.55%)	2.18(1.40%)
TS8	S6	Body	(0.024, 0.126)	(0.019, 0.077)	3.90(7.73%)	1.97(3.911%)

trajectories are well aligned over a satellite or a floorplan image. We adjusted the scales (meters per pixel) based on the rulers, and manually specified the starting point and the initial orientation.

Parameter λ: Table 3 shows the impact of the parameter λ in Equation 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 on its own. From this result, we set $\lambda = 0.1$ as our default parameter value.

6.2. Velocity evaluations

Our cascaded velocity regression achieves the mean squared errors of 0.016 [m^2/s^2] and 0.015 [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 94.70% and 93.65%, respectively. Lastly, we have evaluated the all-in-one regression model by SVR without the placement classification. The mean squared errors on the X and Z axes are 0.32 [m^2/s^2] and 0.54 [m^2/s^2], respectively, which are much 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, while 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 7 shows that our system generalizes reasonably well under all placement types, in particular, still keeping the mean positional errors below 3%.

Unseen subjects: The last experiment evaluates the generalization capability to unseen subjects (marked as S5 and S6). 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 one sub-

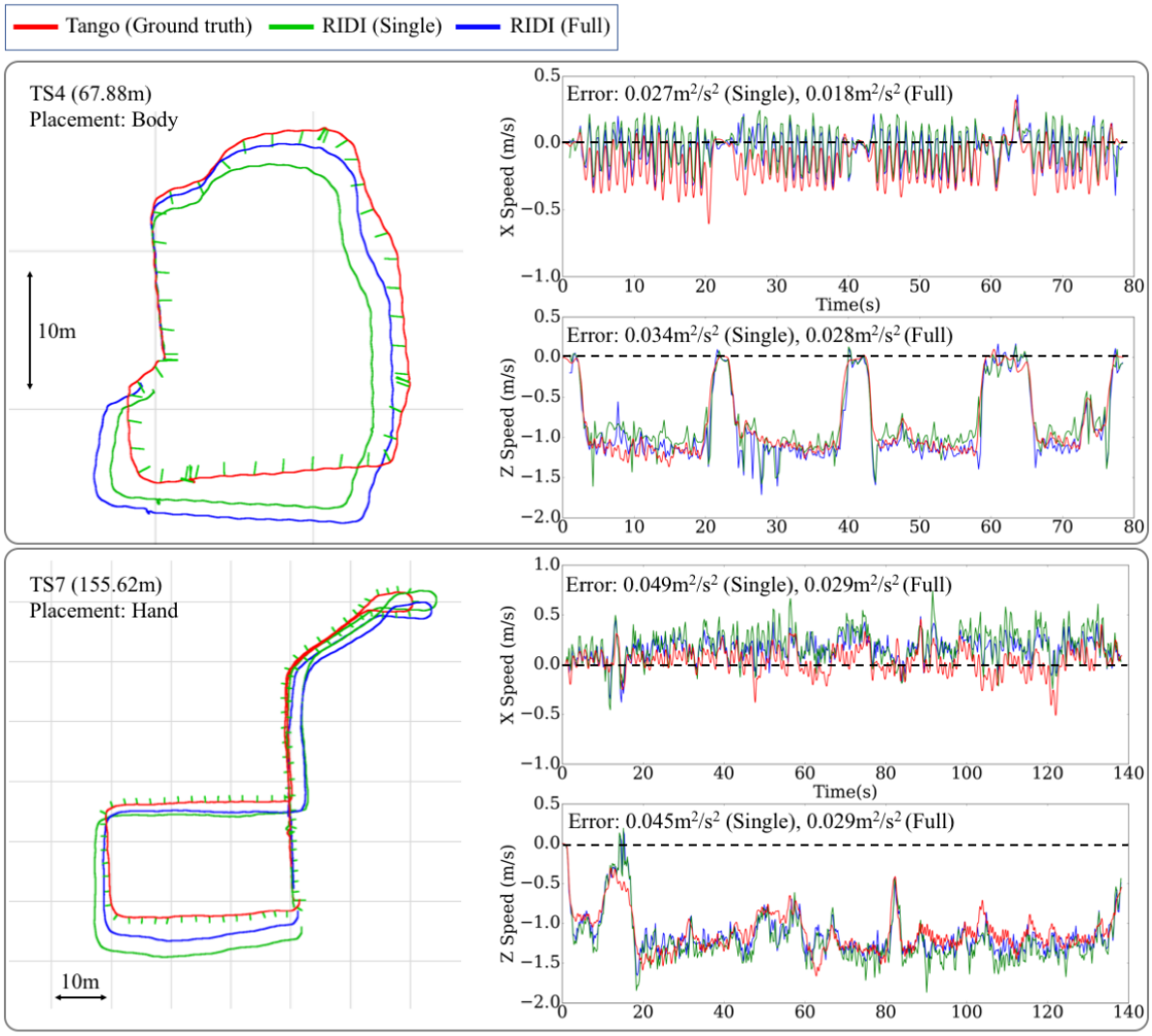


Figure 8: Generalization to unseen subjects. We varied the number of human subjects in the training data and evaluated two RIDI models for unseen testing subjects. RIDI (single) uses training data only from one subject, while RIDI (Full) uses training data from the four subjects.

ject (S1). RIDI (Full) is trained on data from the four subjects (S1-S4). For fair comparisons, we have down-sampled both training sets to 22,000 samples. Figure 8 and Table 4 demonstrate that the Full model generalizes well, in particular, below 5% MPE in most cases. However, the system performs worse in some sequences, for example, 8 meter positional errors after 100 meters of walking, which is too large for many indoor applications. Another important future work is to push the limit of the generalization capability by collecting more data and designing better regression machineries.

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 any-

body 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.

8. Acknowledgement

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

References

[1] Apple arkit. https://developer.apple.com/arkit/. 2

[2] Eigen 3.3.2. http://eigen.tuxfamily.org/. 4

[3] Google arcore. https://developers.google.com/ar/. 2

[4] Google project tango. https://get.google.com/tango/. 2

[5] Microsoft hololens. https://www.microsoft.com/microsoft-hololens/en-us. 2

[6] Opencv 3.2. http://opencv.org/. 4

[7] S. Agarwal, K. Mierle, and Others. Ceres solver. http://ceres-solver.org. 4

[8] P. Bahl and V. N. Padmanabhan. Radar: An in-building rf-based user location and tracking system. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 2, pages 775–784. Ieee, 2000. 2

[9] 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*, pages 225–234. ACM, 2013. 2

[10] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on Robotics*, 32(6):1309–1332, 2016. 2

[11] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *arXiv preprint arXiv:1607.02565*, 2016. 2

[12] B. Ferris, D. Fox, and N. Lawrence. Wifi-slam using gaussian process latent variable models. In *In Proceedings of IJCAI 2007*, pages 2480–2485, 2007. 2

[13] Google. Google cardboard. https://vr.google.com/cardboard/. 2

[14] J. Huang, D. Millman, M. Quigley, D. Stavens, S. Thrun, and A. Aggarwal. Efficient, generalized indoor wifi graphslam. In *IEEE International Conference on Robotics and Automation*, pages 1038–1043, 2011. 2

[15] P. Kasebzadeh, C. Fritsche, G. Hendeby, F. Gunnarsson, and F. Gustafsson. Improved pedestrian dead reckoning positioning with gait parameter learning. In *Information Fusion (FUSION), 2016 19th International Conference on*, pages 379–385. IEEE, 2016. 2

[16] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *ISMAR*, pages 225–234. IEEE, 2007. 2

[17] F. Li, C. Zhao, G. Ding, J. Gong, C. Liu, and F. Zhao. A reliable and accurate indoor localization method using phone inertial sensors. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 421–430. ACM, 2012. 2

[18] C.-H. Lim, Y. Wan, B.-P. Ng, and C.-M. S. See. A real-time indoor wifi localization system utilizing smart antennas. *IEEE Transactions on Consumer Electronics*, 53(2), 2007. 2

[19] A. Mansur, Y. Makihara, R. Aqmar, and Y. Yagi. Gait recognition under speed transition. In *CVPR*, pages 2521–2528, 2014. 2

[20] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, 2015. 2

[21] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison. Dtam: Dense tracking and mapping in real-time. In *ICCV*, pages 2320–2327. IEEE, 2011. 2

[22] Samsung. Samsung gear vr. http://www.samsung.com/global/galaxy/gear-vr/. 2

[23] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):199–222, 2004. 3

[24] T. von Marcard, B. Rosenhahn, M. J. Black, and G. Pons-Moll. Sparse inertial poser: Automatic 3d human pose estimation from sparse imus. In *Computer Graphics Forum*, volume 36, pages 349–360. Wiley Online Library, 2017. 2

[25] X. Yun, E. R. Bachmann, H. Moore, and J. Calusdian. Self-contained position tracking of human movement using small inertial/magnetic sensor modules. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2526–2533. IEEE, 2007. 2

[26] Q.-Y. Zhou and V. Koltun. Simultaneous localization and calibration: Self-calibration of consumer depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 454–460, 2014. 3