Cortes Reina, Santiago; Solin, Arno; Kannala, Juho

Deep learning based speed estimation for constraining strapdown inertial navigation on smartphones

Published in:
2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP)

DOI:
10.1109/MLSP.2018.8516710

Published: 01/01/2018

Document Version
Peer reviewed version

Please cite the original version:
Cortes Reina, S., Solin, A., & Kannala, J. (2018). Deep learning based speed estimation for constraining strapdown inertial navigation on smartphones. In 2018 IEEE 28th International Workshop on Machine Learning for Signal Processing (MLSP) (pp. 1-6). [8516710] Aalborg: Institute of Electrical and Electronics Engineers. https://doi.org/10.1109/MLSP.2018.8516710

DEEP LEARNING BASED SPEED ESTIMATION FOR CONSTRAINING STRAPDOWN INERTIAL NAVIGATION ON SMARTPHONES

Santiago Cortés *Arno Solin* *Juho Kannala*

Aalto University
Department of Computer Science
Espoo, Finland

ABSTRACT

Strapdown inertial navigation systems are sensitive to the quality of the data provided by the accelerometer and gyroscope. Low-grade IMUs in handheld smart-devices pose a problem for inertial odometry on these devices. We propose a scheme for constraining the inertial odometry problem by complementing non-linear state estimation by a CNN-based deep-learning model for inferring the momentary speed based on a window of IMU samples. We show the feasibility of the model using a wide range of data from an iPhone, and present proof-of-concept results for how the model can be combined with an inertial navigation system for three-dimensional inertial navigation.

***Index Terms*—** Inertial navigation, deep learning, smartphone data

1. INTRODUCTION

Recently, there has been a growing interest towards implementing inertial navigation systems (INS, [1, 2]) on smart mobile devices [3, 4, 5]. Indeed, most current smart devices, like phones, tablets and watches are equipped with inertial measurement units (IMU) implemented as microelectromechanical systems (MEMS). These sensors are typically used for orientation estimation and gravitation tracking [6] or activity classification, but their use for inertial navigation is challenging due to the low quality and high noise levels of the sensors, because errors accumulate rapidly in the double integration process that is required to get position from accelerations. Yet, there would be plenty of applications and use cases, ranging from indoor navigation to augmented reality and robotics, which would benefit from full six degree of freedom inertial navigation using cheap MEMS based sensors.

Due to the above challenges, there are not yet accurate INS based tracking approaches that would be suitable for handheld smart devices in the general case. However, in some constrained use cases, such as the case of foot-mounted IMUs [7], inertial navigation may provide good results thanks to the frequent zero-velocity updates that can be detected automatically when the sensor is at rest [8]. Also, if the IMU can be combined with a video camera, then visual-inertial odometry (VIO) methods can provide good accuracy [9, 10]. However, in many use cases VIO is not a possible solution since capturing and processing video causes highly increased battery usage. In addition, it requires unobstructed field of view to an environment that has enough distinguishable visual features. Most VIO methods are not robust to full occlusions.

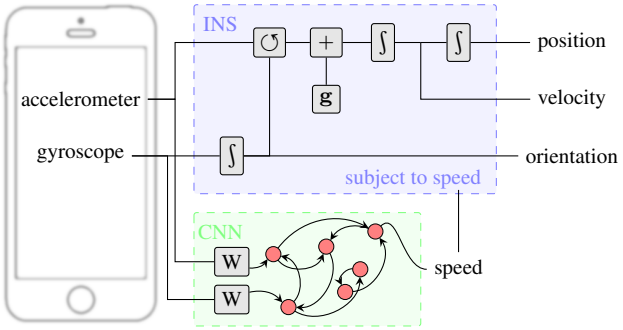


Fig. 1: Sketch of our approach. A convolutional network is used to regress speed information which is used to softly constrain an INS system.

If foot-mounted sensors or cameras are not available, most inertial sensor based approaches for pedestrian dead-reckoning (PDR) rely on step counting [11, 12, 13]. These methods typically assume horizontal 2D motion and are not applicable for wheeled motion or tracking pedestrians in 3D environments or in escalators and elevators. Moreover, if the device is handheld so that the sensor orientation is not fixed with respect to the walking direction, then heading estimation is an additional difficulty besides step detection and even the most robust current systems (*e.g.* [12]) have limited performance in challenging use cases as shown by [5].

In this paper, two different speed concepts are discussed: *Momentary speed* is defined as the average speed over a short window in time. *Instantaneous speed* is defined as the norm of the velocity vector in the state at a given point in time.

Very recently, there have been efforts to utilize machine learning for regressing the momentary horizontal velocity vector from a short time window of gyroscope and accelerometer data [4, 5]. The ground truth motion trajectories for training are obtained with an optical motion capture system (Vicon) [5] or using visual-inertial odometry (Google Tango) [4]. In [4] velocity is estimated using support vector regression whereas [5] uses recurrent neural networks (LSTM). Both of the aforementioned approaches train a separate regressor per use case (*e.g.* phone in hand, bag, pocket or on trolley) and [4] also trains a support vector classifier which first detects the use case (but is error-prone in practice). Since both approaches only consider horizontal planar motion and performance is not evaluated thoroughly in scenarios where different use cases are mixed (*e.g.* phone both in hand and pocket during the same session), it is clear that the existing methods do not yet provide a general and practical six degrees of freedom inertial navigation solution for smart devices.

In this paper we take a different approach and, instead of trying to solve the inertial navigation problem end-to-end using neural networks like [5], we aim at combining machine learning based speed estimation with a classical method [3], which is based on probabilistic sensor fusion using extended Kalman filtering (EKF). That is, we build upon recent work [3], which has shown that utilizing automatic zero-velocity updates (ZUPTs) and pseudo-measurements for limiting momentary speed can give accurate trajectory estimates in varying use cases. However, often in free handheld movement ZUPTs can not be established frequently enough to constrain motion sufficiently. Hence, in this work we aim at using machine learning to improve the accuracy of the momentary speed estimates which are used as pseudo-measurements in [3]. Thus, instead of assuming a globally constant momentary speed with high uncertainty like in [3], we predict more accurate data-driven momentary speed estimates that can be used as pseudo-measurements with smaller uncertainty (see Fig. 1). It should be noted that our approach is not limited to 2D motion like [4, 5], as we regress the scalar speed and not a 2D velocity vector.

The contributions of this paper are:

- A novel neural network model for regressing bounds of speed from short time windows of IMU information, based on recognition of the motion pattern.
- Experimentally showing that a single regression model can provide accurate speed estimates in varying use cases involving handheld motion while walking, standing or traveling in elevators and escalators.
- A versatile dataset with visual-inertial odometry based ground truth for training models on inertial only data.
- Proof-of-concept results showing that the speed estimates provide additional benefits in data-driven inertial navigation for orientation free 3D odometry.

This paper is structured as follows. In section 2, we first introduce the problem formulation, then focus on the speed regression subproblem. Then we present the training data collected on a smartphone and go through the pre-processing steps. in section 3, we present results for the speed regression subproblem (which is the main focus) and then show proof-of-concept results for the constrained INS system in challenging settings. Finally, the model and results are discussed.

2. MATERIALS AND METHODS

A sketch of our approach is given in Figure 1. As inputs we use the three-axis gyroscope and accelerometer data from a smartphone. This data is passed to an inertial navigation system (labeled 'INS' and adopted following [3]) doing statistical inference on the current 3D position, velocity, and orientation. The blocks '$\int$', '$\circlearrowleft$', and '$\mathbf{g}$' denote integration, rotation, and gravity, respectively. Our main interest is, however, in the 'CNN' block which takes in a window (labeled with the windowing operator 'W') of IMU data and tries to infer the momentary scalar speed to be used as a constraint in the INS block.

Given a mode of locomotion, the momentary speed of an agent over a couple of seconds is usually well constrained. In the case of human walking, the momentary speed depends on the person, the terrain and the particular gait. Even in these highly variable cases, there are reasonable bounds for the speed. Those reasonable bounds are introduced into a Kalman filter in the form of a pseudo-velocity update, a weakly informative measurement on the first derivative of the position.

The raw IMU data contains enough information to calssify the motion into high level gait and transportation classes [4]. Instead of classifying into labeled classes, we directly regress the momentary speed.

For this we use a small convolutional network that regresses the speed given a two second window of IMU data at 100 Hz. The ground truth velocity is taken as the displacement over the window as reported by ARKit divided by the time difference. This, in theory, is the average of all the instantaneous speeds over the window.

2.1. Speed regression

The estimation is performed by a convolutional neural network (see, *e.g.*, [14]) tasked with regressing the average speed over a finite window of measurements. The hypothesis is that given enough training on a locomotion mode, the momentary speed can be regressed. The input of the network is the six channels of inertial information over a few seconds and the output is the norm of the velocity vector.

The sample windows have a random starting point, which means that the patterns and relations that enable the classification are randomly shifted in the signal. This problem is

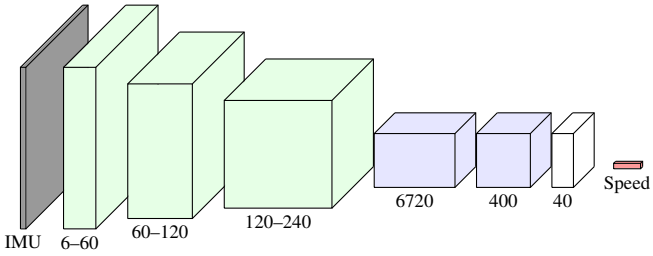


Fig. 2: Structure of the network used, convolutional layers (green) show the number of input–output channels, while fully connected layers (blue) show the number of units. All layers except the output (white) have rectified linear activation (RELU).

naturally solved with a convolutional network which is by design invariant to shifts. Once the features are read by the convolutional layers, the fully connected layers perform the regression of the speed.

The structure of the network and the per layer parameters are shown in Figure 2. All convolutional layers have kernel length 10 and stride of 1. Since the network has a relatively small size, no pooling layers are used. The cost function is the squared error of the momentary speed,

$$\mathcal{L} = (S_{\text{pred}} - \|\mathbf{p}_T - \mathbf{p}_0\|/T)^2, \tag{1}$$

where S_{pred} is the predicted speed, $\mathbf{p}_0 \in \mathbb{R}^3$ and $\mathbf{p}_T \in \mathbb{R}^3$ are respectively the first and last positions of the window and T is the time between the position samples.

The network was trained for 2000 epochs, using mini batches of size 10. Optimization is performed with the Adam algorithm [15]. It was implemented in pytorch (https://pytorch.org/) with pandas for data management. Training took roughly seven seconds per epoch in CPU-only training.

2.2. Data acquisition

For capturing (see details on data acquisition in [16]) authentic sensor data from a smartphone, we implemented (in Swift 4) a data capture application for an Apple iPhone 6s (Apple Inc., https://www.apple.com/iphone-6s). The application captures raw sensor data from the built-in phone three-axis accelerometer and gyroscope (through the CoreMotion API). Data samples are acquired at 100 Hz and timestamped by the platform.

For training, we capture phone odometry information from the Apple ARKit (https://developer.apple.com/arkit/) visual-inertial framework, which runs information fusion of the camera and IMU data for inferring the six degrees-of-freedom relative motion of the device. ARKit can provide a low-drift movement trajectory with associated orientation information. The ARKit output is acquired simultaneously and time-locked to the sensor data output at 60 Hz. Additionally,

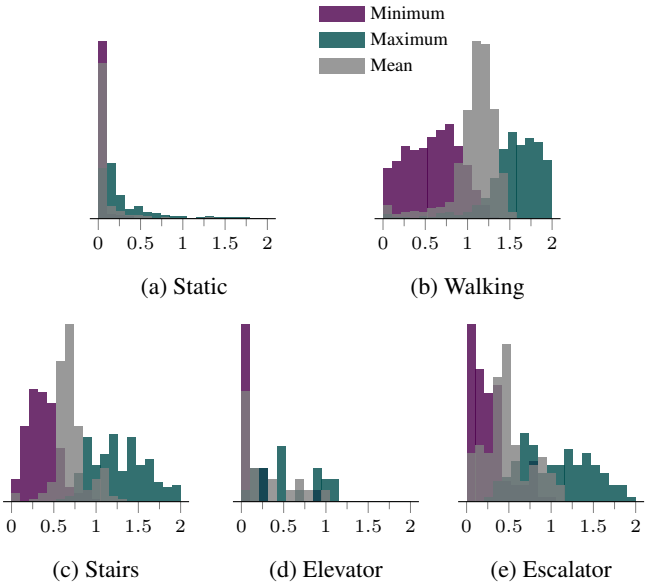


Fig. 3: Histograms of **minimum**, **maximum**, and **mean** instantaneous speed (m/s) for each window in the labeled modes of locomotion.

for reference we also save the video stream of the back-facing iPhone camera at 60 fps.

The training and testing data was captured in various environments by walking around with the mobile phone, primarily indoors on campus and in a shopping mall. The data also features outdoor sequences. The parts of the data featuring movement in stairs and escalators or standing still while holding the device, were manually annotated as a post-processing step (using the reference video stream).

The length of the training data is approximately 100 minutes and it contains some 4.5 km of movement (primarily walking). The example test sequence is a continuous data set of length 391 s / 345 m which includes planar walking, stairs and standstill.

2.3. Pre-processing

The IMU measurements are in the phone coordinate frame, the ARKit position is in a global coordinate frame. For the scalar speed, this discrepancy is not a problem, but for more sophisticated measurements, the discrepancy should be resolved. The orientation of the ARKit coordinate frame can be estimated from the gravity vector.

For training, we perform 10-fold cross-validation, where the data is randomly split into 90% training and 10% validation sets. A separate test sequence is held out and not used in training.

The sequences contain normal walking, standing still, and movement in staircases, escalators and elevators. Given that the dataset is based on natural sequences, there are much more

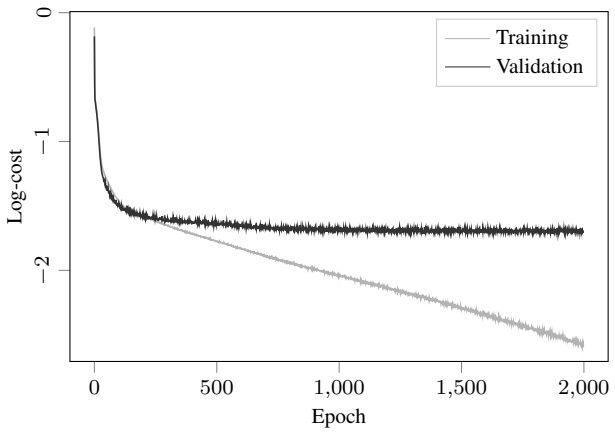


Fig. 4: Log-cost progression by epoch during training for one of the folds. As the training progresses the cost stabilizes for the validation (held-out) set, indicating convergence.

training samples for walking than for the other modes.

The entire dataset has been labeled with the different motion modes. The labels are not used in training, only in visualization of the results. Since it is hard to strictly define the modes, they were manually annotated using subjective analysis of the video. The result still illuminates on the performance and shortcomings.

The histograms of the instantaneous speed of each of the classes can be seen in Figure 3. The distribution of maximum and minimum speeds shows that there is a lot of variation on each frame, but the mean remains relatively constrained.

3. EXPERIMENTS

We present experimental validation primarily for estimating the speed, but also show proof-of-concept results for using the speed estimates for improving odometry estimates.

3.1. Network performance

The CNN network was validated with 10-fold cross-validation, the average RMSE on the validation data is 0.13 m/s. The result of the 10-fold cross-validation is plotted against the ground truth in Figure 5.

Green dots show samples labeled as walking. Walking motion is the bulk of the training data. Blue dots show the samples marked as static. Static is defined as not in the process of going from one place to another—this includes standing and short 'looking around' movements. For these two modes the predictions agree well with the ground truth. During standstill slight leaking towards higher speeds can be noticed. Red squares show samples labeled as stairs. Stairs samples include the short walking samples in between staircases. Dark gray dots show samples labeled as escalator. Just as with stairs, these include the short walks between escalators.

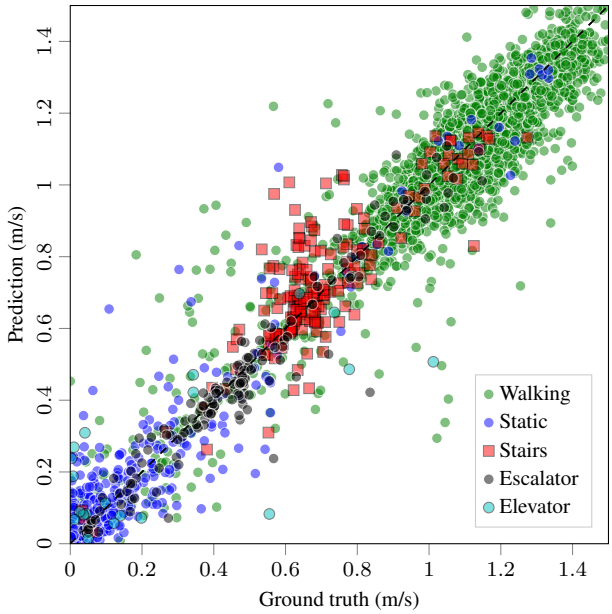


Fig. 5: Labeled scattered validation results for 10-fold cross-validation. Ideally all cases would fall on the identity line. The labels were not used during training; they are just for interpreting the results.

Finally, cyan dots show the few elevator examples, these were recorded in a glass-walled elevator in order to keep the functionality of the ARKit.

The network was also tested with a newly recorded sequence on the same device that contains walking, standing still, and going up and down stairs. The whole ground truth speed and prediction by the network can be seen in Figure 6. The sequence is sampled every second, so the windows have a 50% overlap. The resulting RMSE is 0.20 m/s. The background color shows the label for that portion of the test sequence. The colors are the same used in Figure 5.

3.2. Inertial navigation with speed constraints

The relevance of the momentary speed in INS systems is put to the test in a simple experiment. The test dataset used for Figure 7 is put trough an INS based fix-point interpolator where the fix-points are sampled from the position data every 17 seconds. The INS is based on the system presented in [3].

The original system uses zero velocity updates and a constant pseudo-speed measurement to prevent fast drifts that are common in these navigation systems. In this case the pseudo-measurement is replaced with the regressed speed and the measurement noise is reduced to more accurately reflect the knowledge gained from the CNN. The pseudo-speed measurement is of the form $h_{\text{pseudo}}(\mathbf{x}) = \|\mathbf{v}\|$, where $\mathbf{v}$ is the speed component of the state vector $\mathbf{x}$.

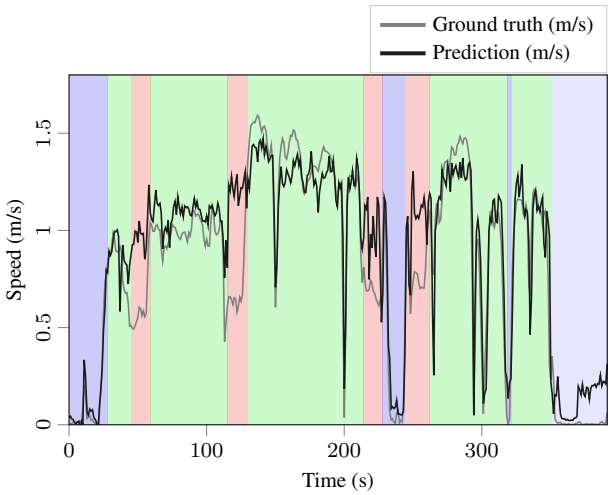


Fig. 6: Speeds from an independent test sequence featuring movement on three floor levels. The background color indicates the motion mode (**static**, **walking**, and **stairs**).

The original INS system uses constant 0.75 m/s speed updates on the L2 norm of the velocity vector as it is stored on the extended Kalman Filter (EKF) state. This maintains the speed at a constrained value. The updated version uses the regressed speed as the pseudo-speed measurement. This is important when the actual speed differs from the 0.75 m/s the most. In this case the measurement is more certain below 0.2 m/s.

Figure 7 shows the result of the test sequence reconstruction using the INS based interpolation. In (d) the whole path is shown, it spans three floors of an office building. Subfigure (a) shows the path interpolated using no speed constrains at all, (b) shows the path interpolated using the constant pseudo-speed (as in [3]), and (c) shows the path interpolated using our CNN regressed speed.

The RMSE error for the reconstructed paths are:

No contraints:	238.38 m
Pseudo-speed:	0.87 m
CNN constraint:	0.62 m

From both the RMSEs and Figure 7 it is clear that the unconstrained estimation scheme diverges, while the speed-contrained ones work rather well. In this case, we gain a clear improvement in terms of RMSE by constraining the system by the estimated speed.

4. DISCUSSION

The scatter plot of labeled data in Figure 5 presents several characteristics of the data. The walking motion is clearly over represented in the training data, however, we posit that walking is the main mode of medium range transportation where smartphone based INS is used. The network performs relatively well all around, even with the small sample size for some modes. The test sequence shows that the speed prediction is very accurate for walking and standing still. The network regresses the speed and has no problem in the transition between modes. However, there are errors in all the staircase sequences. It is probably due to under representation in the training data.

The errors are mostly in small portions of one mode surrounded by other modes, for example a three step stair where there was walking before and after. Most people have a different stride to tackle this kind of obstacle as opposed to a full staircase. The histograms in Figure 3 show how the instantaneous speed is distributed among the different motion modes. Even though the instantaneous speeds are not very compact, the mean speed is fairly consistent within each mode.

Section 3.2 shows how the regressed speed helps a basic INS system by adjusting the pseudo-update drift with the information from the CNN. The low speed pseudo-update covers the space between the ZUPTs. The result can be seen in the first few seconds of the capture where the motion is minimal.

5. CONCLUSION

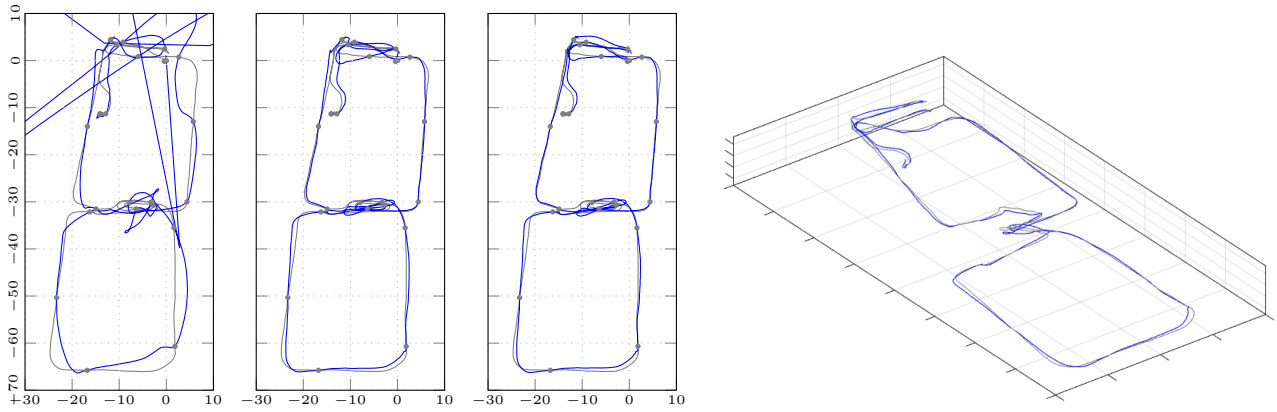
In this paper we have proposed a scheme for free three-dimensional inertial navigation, that combines classical strict physics-driven inertial navigation with a purely data-driven approach of injecting additional knowledge through estimating the momentary speed by a CNN. We see this as a good split between more strict model-based modelling and 'blind' learning from data.

Our main interest was in evaluating the CNN model. We present an approach to regress the momentary speed based on a two second window of IMU data. The data was trained on the displacement as reported by the visual-inertial method ARKit.

The system was tested on a new sequence unknown to the training. The results were accurate on both walking and standing still modes, but could be improved for use on stairs. Finally, in a proof-of-concept study, we used the regressed speed in a functioning INS system to help constrain the movement. We gained improvements in terms of RMSE.

For the speed estimates to be more helpful, there are a number of possible future research directions to consider. In free use cases, kicking off the estimation can be a challenge. Furthermore, considering the speeds in the horizontal and vertical directions separately might help the challenges related to movement in escalators and stairs. How well the methods generalize over devices with different sensor biases should also be tackled by broader sets of training data.

The data and codes for the speed regression problem can be accessed at: https://aaltovision.github.io/deep-speed-constrained-ins/.



(a) No constraints (b) Pseudo-speed (c) CNN constraint (d) Perspective view of movement on three floors

Fig. 7: The reconstructed metric path from the test sequence in Fig. 6. (a)–(c) show the INS results with (a) no speed constraint, (b) a pseudo-constraint, and (c) using the CNN estimated speed. The ARKit ground truth is in gray and fix points shown by dots. The RMSE errors are 238.38 m, 0.87 m, and 0.62 m, respectively.

Acknowledgements. This research was supported by the Academy of Finland grants 308640, 277685, and 295081. We acknowledge the computational resources provided by the Aalto Science-IT project.

6. REFERENCES

[1] Christopher Jekeli, *Inertial Navigation Systems with Geodetic Applications*, Walter de Gruyter, Berlin, Germany, 2001.

[2] Kenneth R Britting, *Inertial Navigation Systems Analysis*, Wiley-Interscience, New York, 2010.

[3] Arno Solin, Santiago Cortes, Esa Rahtu, and Juho Kannala, "Inertial odometry on handheld smartphones," in *Proceedings of the International Conference on Information Fusion (FUSION)*, 2018.

[4] Hang Yan, Qi Shan, and Yasutaka Furukawa, "RIDI: Robust IMU double integration," *arXiv preprint arXiv:1712.09004*, 2017.

[5] Changhao Chen, Xiaoxuan Lu, Andrew Markham, and Niki Trigoni, "IONet: Learning to cure the curse of drift in inertial odometry," in *Proceedings of AAAI Conference on Artificial Intelligence*, 2018.

[6] Simo Särkkä, Ville Tolvanen, Juho Kannala, and Esa Rahtu, "Adaptive Kalman filtering and smoothing for gravitation tracking in mobile systems," in *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Banff, Canada, 2015, pp. 1–7.

[7] Eric Foxlin, "Pedestrian tracking with shoe-mounted inertial sensors," *Computer Graphics and Applications*, vol. 25, no. 6, pp. 38–46, 2005.

[8] John-Olof Nilsson, Amit K. Gupta, and Peter Händel, "Foot-mounted inertial navigation made easy," in *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Busan, Korea, 2014, pp. 24–29.

[9] Arno Solin, Santiago Cortes, Esa Rahtu, and Juho Kannala, "PIVO: Probabilistic inertial-visual odometry for occlusion-robust navigation," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2018.

[10] Thomas Schneider, Marcin Dymczyk, Marius Fehr, Kevin Egger, Simon Lynen, Igor Gilitschenski, and Roland Siegwart, "maplab: An open framework for research in visual-inertial mapping and localization," *IEEE Robotics and Automation Letters*, 2018.

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

[12] Zhuoling Xiao, Hongkai Wen, Andrew Markham, and Niki Trigoni, "Robust pedestrian dead reckoning (R-PDR) for arbitrary mobile device placement," in *Proceedings of the International Conference on Indoor Positioning and Indoor Navigation (IPIN)*, Busan, Korea, 2014, pp. 187–196.

[13] Al Mansur, Yasushi Makihara, Rasyid Aqmar, and Yasushi Yagi, "Gait recognition under speed transition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2521–2528.

[14] Yann A LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, R.E. Howard, Wayne E Hubbard, and Lawrence D Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[15] Diederik P. Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *Proceedings of the 3rd International Conference for Learning Representations (ICLR)*, San Diego, CA, 2015.

[16] Santiago Cortés, Arno Solin, Esa Rahtu, and Juho Kannala, "ADVIO: An authentic dataset for visual-inertial odometry," in *Proceedings of the European Conference on Computer Vision (ECCV)*, Munich, Germany, 2018.