

Neural Inertial Localization

Sachini Herath^{1*} David Caruso² Chen Liu² Yufan Chen² Yasutaka Furukawa¹

¹Simon Fraser University, BC, Canada ²Reality Labs, Meta, Redmond, USA

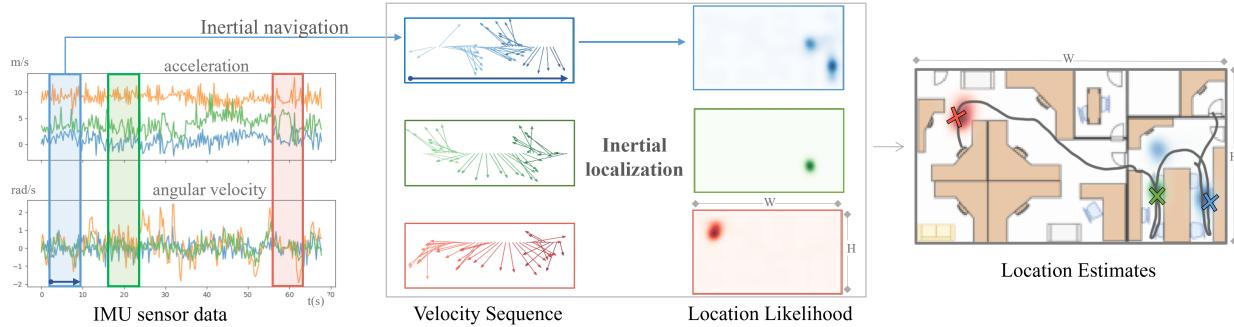


Figure 1. From IMU measurements to location estimates. Given inertial sensor data (left), our approach (center) uses a neural inertial navigation technique to find a sequence of velocity vectors; then train a scene-specific transformer-based neural architecture, which maps a velocity sequence to a location likelihood. The figure shows sample localization results from our office dataset. Color shows temporal correspondence between input(left), intermediate representations(middle), and location (right).

Abstract

This paper proposes the inertial localization problem, the task of estimating the absolute location from a sequence of inertial sensor measurements. This is an exciting and unexplored area of indoor localization research, where we present a rich dataset with 53 hours of inertial sensor data and the associated ground truth locations. We developed a solution, dubbed neural inertial localization (NILoc) which 1) uses a neural inertial navigation technique to turn inertial sensor history to a sequence of velocity vectors; then 2) employs a transformer-based neural architecture to find the device location from the sequence of velocities. We only use an IMU sensor, which is energy efficient and privacy preserving compared to WiFi, cameras, and other data sources. Our approach is significantly faster and achieves competitive results even compared with state-of-the-art methods that require a floorplan and run 20 to 30 times slower. We share our code, model and data at <https://sachini.github.io/niloc>.

1. Introduction

Imagine one stands up, walks for 3 meters, turns right, and opens a door in an office. This information might be sufficient to identify the location of the individual. A recent

breakthrough in inertial navigation [10, 14, 22] allows us to obtain such motion history using an inertial measurement unit (IMU). What is missing is the technology that maps a motion history to a location. This paper addresses this gap, seeking to open a new paradigm in the localization research, named “inertial localization”, whose task is to infer the location from a sequence of IMU sensor data.

Indoor localization is a crucial technology for location-aware services, such as mobile business applications for consumers, entertainment (e.g., Pokemon Go) for casual users, and industry verticals for professional operators (e.g., maintenance at a factory). State-of-the-art indoor localization systems [5] mostly rely on WiFi, whose infrastructure is ubiquitous thanks to the demands on Internet of Things (IoT). Nevertheless, accuracy of WiFi based localization depends on infrastructure (i.e. number of access points) thus cannot scale easily to non-commercial private spaces.

IMU is a powerful complementary modality to WiFi, which has proven effective for the navigation task recently [10, 14, 22]. IMU 1) works anytime anywhere (e.g., inside a pocket/bag/hand); 2) is energy efficient to be an always-on sensor and 3) protects the privacy of bystanders.

This paper introduces a novel inertial localization problem as a task of estimating the location from a history of IMU measurements. The paper provides the first inertial localization benchmark, consisting of 53 hours of motion data and ground-truth locations over 3 buildings. The

*Corresponding author sherath@sfu.ca. (This work was partially done when Sachini was an intern at Meta.)

paper also proposes an effective solution to the problem, dubbed neural inertial localization (NILoc). NILoc first uses a neural inertial navigation technique [10] to turn IMU sensor data into a sequence of velocity vectors, where the remaining task is to map a velocity sequence to a location. The high uncertainty in this remaining task is the challenge of inertial localization. For instance, a stationary motion can be anywhere, and a short forward motion can be at any corridor. To overcome the uncertainty, our approach employs a Transformer-based neural architecture [27] (capable of encoding complex long sequential data) with a Temporal Convolutional Network (further expanding the temporal capacity by compressing the input sequence length) and an auto-regressive decoder (handling arbitrarily long sequential data).

The contributions of the paper are 3-fold: 1) a novel inertial localization problem, 2) a new inertial localization benchmark, and 3) an effective neural inertial localization algorithm. We will share our code, models and data.

2. Related Work

2.1. Indoor localization

Outdoor navigation predominately uses satellite GPS. Indoor localization often relies on multiple data sources such as images, WiFi, magnet, or IMU. We review indoor localization techniques based on the input modalities.

Image based localization estimates a camera’s 6DoF pose from a query image. A classical approach is to detect feature pixels, establish 2D-to-3D or 2D-to-2D correspondences, and solve a Perspective-n-Point (PnP) problem [23]. The surge of deep learning allows us to learn these steps by an end-to-end network [30]. Another family of neural architectures directly regress pose parameters [13]. InLoc [24], a system for image-based indoor localization, reports localization error below 1 meter for 69.9% of queries. While being precise, the image modality suffers from a few major drawbacks for serving mobile applications: a camera needs a direct line of sight, consumes significant amount of battery, and reveals information about bystanders.

Wireless localization based on WiFi or Bluetooth is the current main-stream modality for indoor localization [3, 21, 29]. Wireless receivers work anytime anywhere and WiFi infrastructure is ubiquitous thanks to the ever-growing IoT market demands. The wireless modality is not as precise as the image-based approach and reports a minimum 10 meter error radius [9]. This paper studies inertial localization as an effective complementary modality.

Activity and magnetic-fields are other modalities for indoor localization. Activity recognition based on IMU sensor data provides cues on the locations through

a predetermined mapping between activity types and locations [12, 31]. Location specific magnetic-field distortions can also be used to build a localization system through a site survey [1, 18, 26].

IMU and floorplan fusion allows classical filtering methods (e.g., particle filter) to perform localization [25] by using inertial navigation to propagate particles and the floorplan to re-weight particles. This approach is sensitive to cumulative sensor errors in inertial navigation. Correlation between a short motion history (five seconds) and floorplan can provide additional prior to weigh particles [16] but requires start location and orientation to initialize the system. We employ a novel Transformer-based neural architecture that regresses the location from long motion history even under severe bending. Our approach does not require a floorplan, which often misses transient objects (e.g., chairs/desks) and needs occasional updates, thus provide a compelling alternative.

2.2. Inertial navigation

Inertial navigation estimates relative motions from IMU sensor data, that is, linear accelerations from an accelerometer and angular velocities from a gyroscope. Deep learning has made significant progress in recent years, producing accurate 3DoF [10] and 6DoF [14, 22] trajectories by learning repetitive human body motions with Convolutional Neural Network (CNN) [8] or LSTM [11]. The main source of error is the bias in consumer-grade gyroscopes, accumulating into significant orientation errors, which becomes as large as 20 degrees with 5 minutes of motions even after the calibration. Our inertial localization algorithm first uses inertial navigation to estimate a sequence of velocity vectors from IMU data.

3. Inertial Localization Problem

Inertial localization is the task of estimating the location of a subject in an environment, solely from a history of IMU sensor data. There is a training phase and a testing phase i.e. without a use of a floorplan or external location information. During testing, an input is a sequence of acceleration (accelerometer), angular velocity (gyroscope), and optionally magnetic field (compass) measurements, each of which has 3 DoFs. An output is position estimations for a given set of timestamps, when ground-truth positions are available. During training, we have a set of input IMU sensor data and output positions.

Metrics: The localization accuracy is measured by 1) the ratio (%) of correct position estimations within a distance threshold (1, 2, 4, or 6 meters) [24]; and 2) the ratio (%) of correct velocity directions within an angular threshold (20 or 40 degrees). The position ratio is the main metric, while the direction ratio measures the temporal consistency.

Building	Environment		Full Dataset			Test set	
	Dimensions [m ²]	resolution [pixels/m]	#T (#S)	duration [h]	length [km]	By sequence #T (avg. [min])	By length #T (100m)
University A	62.8 × 84.4	2.5	151 (52)	25.57	65.35	25 (12.07)	75
University B	57.6 × 147.2	2.5	91 (3)	14.64	56.93	20 (12.28)	60
Office C	38.4 × 11.2	10.0	81 (1)	12.91	21.36	12 (15.48)	36

Table 1. Inertial localization dataset consists of two university buildings and one office space. The table shows the number of trajectories (#T), the number of subjects in data collection (#S) and the length statistics for full dataset and test sets.

Re-localization task extension: We propose an inertial re-localization task, which is different from inertial localization in that the position \mathbb{R}^2 (and optionally the motion direction $SE(2)$) is known apriori. The task represents a scenario where one uses WiFi to obtain a global position once in a few minutes, while re-localizing oneself in-between with an IMU sensor for energy efficiency.

4. Inertial Localization Dataset

We present the first inertial localization dataset, containing 53 hours of motion/trajecotry data from two university buildings and one office space. Table 1 summarizes the dataset statistics, while Fig. 2 visualizes all the ground-truth trajectories overlaid on a floorplan. Each scene spans a flat floor and the position is given as a 2D coordinate without the vertical displacement. If available, a floorplan image is provided for a scene for qualitative visualization, which depicts architectural structures (e.g., walls, doors, and windows) but does not contain transient objects such as chairs, tables, and couches.

Data collection: We collect IMU sensor data and ground-truth locations with smartphones. In the future, AR devices (e.g., Aria glasses by Meta, Spectacles by Snap) will allow collection of ego-centric datsets with tightly coupled IMU and camera data. We used two devices in this work; 1) a handheld 3D tracking phone (Google Tango, AsusZenfone AR) with built-in Visual Inertial SLAM capability, producing ground-truth relative motions, where the Z axis is aligned with the gravity; and 2) a standard smartphone, recording IMU sensor data under natural phone handling (e.g. in a pocket, hand or used for calling etc.). We utilize Tango Area Description Files [6] to align ground-truth trajectories to a common coordinate frame then manually align with the floorplan. University A contains data from RoNIN dataset [10] aligned manually to a floorplan. Both IMU sensor data and ground-truth positions are recorded at 200Hz.

Test sequences: We randomly select one sixth of the trajectories as the testing data, whose average duration is 13.3 minutes. We also form a set of short fixed-length sub-sequences for testing by randomly cropping three sub-sequences (100 meters) from each testing sequence.



Figure 2. Inertial localization dataset contains IMU measurements and ground-truth locations based on Visual Inertial SLAM in three buildings. The ground-truth trajectories are shown here with random colors, overlaid on the respective floorplans. (Image dimensions are in meters).

NIloc dataset is de-identified to mask the identity of subjects and does not contain any image or video data.

5. NIloc: Neural Inertial Localization

Instead of regressing locations from IMU measurements, our system NIloc capitalizes on neural inertial navigation technology [10] that turns a sequence of IMU sensor data to a sequence of velocity vectors, where our core task will be to turn velocity vectors into location estimations.¹

¹RoNIN ResNet model estimates velocities at the frequency of IMU data. To handle periods of no to little motions (e.g., sitting down), we resample velocities based on the distance of travel. Concretely, we add-up velocity vectors until its length is more than the distance equivalent to one pixel in the location map, then sample one aggregated vector.

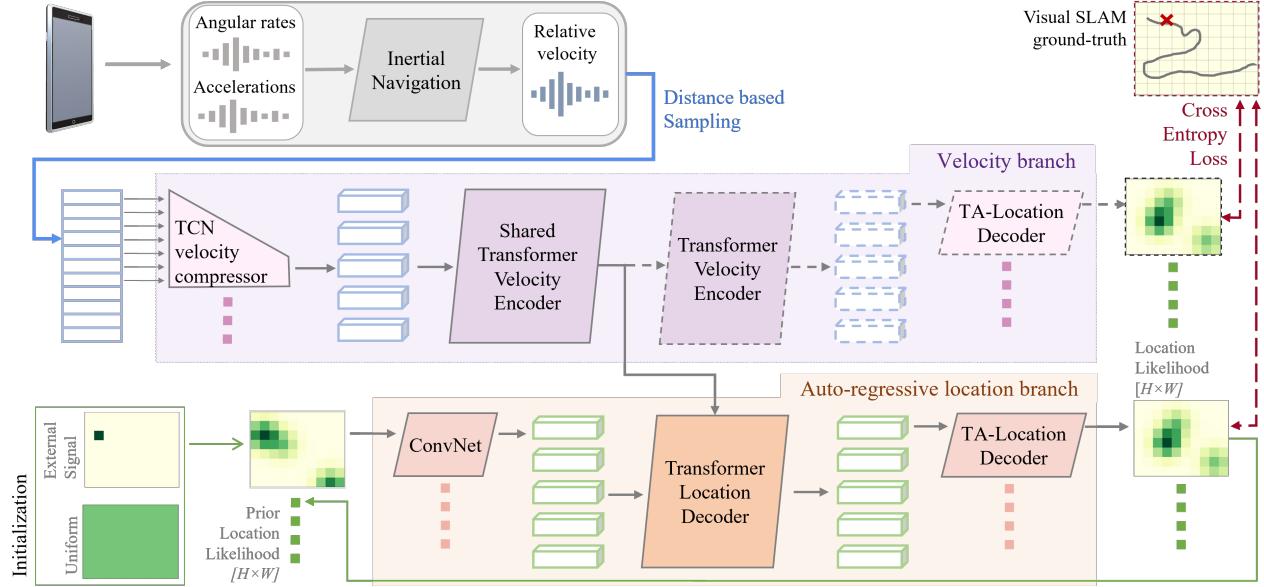


Figure 3. Neural inertial localization system diagram. We use two branch transformer architecture to estimate location likelihood from velocity input. The paths only used in training are shown as dashed lines.

High uncertainty is the challenge in the task. NILoc employs a neural architecture with two Transformer-based network branches [27], capable of using long history of complex motion data to reduce uncertainty. The “velocity branch” encodes a sequence of velocity vectors, where a Temporal Convolutional Network compresses temporal dimension to further augment the temporal receptive field. The “auto-regressive location branch” encodes a sequence of location likelihoods, capable of auto-regressively producing location estimates on a long horizon. The network is trained per scene given training data.

The section explains the two branches (Secs. 5.1 and 5.2), the training scheme (Sec. 5.3), and the data augmentation process (Sec. 5.4), which proves effective in the absence of sufficient training data.

5.1. Velocity branch

The branch estimates a location sequence using a history of velocity data. It consists of three network modules: TCN-based velocity compressor, Transformer velocity encoder, and Translation-aware location decoder.

TCN-based velocity compressor: Transformer is powerful but memory intensive. We use a temporal convolutional network (TCN) [2] to compress a velocity sequence length by a factor of 10, allowing us to process longer motion history. In particular, we use a 2-layer TCN with a receptive field of 10 to compress a sequence of 2D velocity vectors $\{v_t\}$ of length T into a sequence of d -dimensional² feature vectors $\{v'_t\}$ of length $T/10$:

²Dimension d is set to 288, 470, and 448 for buildings A, B, and C, respectively to be proportional to their floor-areas and resolution.

$$\{v_1, v_2, \dots, v_T\} \longrightarrow \{v'_1, v'_2, \dots, v'_{T/10}\}.$$

Transformer encoder: Transformer architecture [27] takes compressed velocity vectors $\{v'_t\}$ as tokens and initializes each feature vector f_t by concatenating the $d/2$ dimensional trigonometric position encoding of the frame index:

$$f_t = [v'_t, \{\cos(w_i t)\}, \{\sin(w_i t)\}]$$

$$w_i = \exp \frac{-\log(10000) * i}{d'} \quad (i = 1, 2, \dots, d/4)$$

f_t is of dimension d' ($= \frac{3}{2}d$). An output embedding e_t per token is also a d' dimensional vector encoding the location likelihood. The encoder has two blocks of self-attention networks. Each block has 2 standard transformer encoder layers with 8-way multi-head attention. Feature vectors after the first block are also passed to the other branch (i.e., auto-regressive location branch).

Translation-aware location decoder: The last module operates on each individual embedding e_t . First, e_t is rearranged into an image feature volume (3D tensor³) and up-sampled via a 3-layer fully convolutional decoder with transpose convolutions. The last layer is a “translation-aware” 1×1 convolution, whose parameters are not shared across pixels. To account for uncertainty, the output location is represented as a 2D likelihood map L_t of size $W \times H$: $L(x, y)$.⁴ This translation-aware layer allows the

³The dimensions (width,height,channels) are 24x18x1, 16x44x1, and 14x48x1 for the three scenes A, B, and C, respectively

⁴The map-extent is determined for each scene by the axis-aligned bounding box of ground-truth locations. We choose the resolution (pixels per meter) so that total number of pixels is around 3 million (See Table 1).

network to easily learn translation-dependent information such as “people never come to this location” or “one always pass through this doorway”.

5.2. Auto-regressive location branch

The location branch combines the velocity features from the velocity branch and prior location likelihoods, which comes from its past inference or an external position information such as WiFi.

The location branch has the same architecture as the velocity branch with two differences. First, instead of a TCN-based velocity compressor, we use a ConvNet to convert each $W \times H$ likelihood map into a d' -dimensional vector. We use the same trigonometric position encoding (but with dimension d' instead of $d/2$ to match the dimension), which is added to the vector. Second, we inject velocity features from the velocity branch via cross-attention after every self-attention layer (i.e., before every add-norm layer). The rest of the architecture is the same. Note that both branches predict locations, and have different trade-offs (See Sect. 6.4 for ablation study and discussion).

At inference time, we first evaluate the velocity branch in a sliding window fashion to compute velocity feature vectors. The location branch takes a history of location likelihoods up to 20 frames: $\{L_t, L_{t-1}, \dots, L_{t-19}\}$. L_0 encodes external initial location information (e.g., from WiFi) or a uniform distribution if not available. At the output, a node initialized with a likelihood at frame t' will have a likelihood estimate at frame $t' + 1$. Therefore, we infer a likelihood up to 20 times for one frame, where we compute the weighted average as the final likelihood by decreasing the weights from 1.0 down to 0.05 from the first inference result to the last.

5.3. Training scheme

We use a cross entropy loss at both branches. The ground-truth likelihood is a zero-intensity image, except for one pixel at the ground-truth location whose value is 1.0. We employ parallel scheduled sampling [17] to train the auto-regressive location branch without unrolling recurrent inferences. The process has two steps. First, we pass GT likelihoods to all the input tokens and make predictions. Second, we keep the GT likelihoods in the input tokens with probability $r_{teacher}$ (known as a teacher-forcing ratio), while replacing the remaining nodes with the predicted likelihoods. The back-propagation is conducted only in the second step. $r_{teacher}$ is set to 1.0 in the first 50 epochs, and reduced by 0.01 after every 5 epochs.

5.4. Synthetic data generation

The Transformer architecture requires a large amount of training data.⁵ We crop data over different time windows

⁵The COVID pandemic further makes the data collection challenging.

to augment training samples. However, in the absence of sufficient training data, we use the following three steps to generate more training samples synthetically: 1) Compute a likelihood map of training trajectories (i.e., where they pass through); 2) Randomly pick a pair of locations from high likelihood areas; and 3) Solve an optimization problem to produce a trajectory that is smooth and pass through the area of high likelihood. Given a synthesized trajectory, we sample velocity vectors based on the distance of travel as in our preprocessing step, which are directly passed to the TCN-based velocity compressor during training. All the steps are standard heuristics and we refer the details to the supplementary.

6. Experimental Results

6.1. Baseline methods

To our knowledge, no prior work addresses indoor localization from IMU data alone. Therefore, we compare with the following three techniques that fuse IMU and floorplan.⁶ Note that our method is the only *inertial localization* that uses IMU data alone without the floorplan images. We briefly explain the three techniques.

- **Particle Filter (PF)** maintains a set of particles, each of which stores the location, heading direction, and bias/scale error correction terms. Starting from a Gaussian distribution around the given initial location or a uniform distribution otherwise, the system updates the states of the particles based on the inertial navigation result and the floorplan information (i.e., down-weight particles if outside the walkable region). We take the particle closest to the weighted median x/y coordinate as the location prediction.
- **Learned Prior (LP) [16]** is also a particle filter based approach while using deep networks to help update particle weights. A location likelihood, computed by the dot product between floorplan features extracted by UNet [20] and motion features extracted by LSTM [11], is used to weight the particles. We use our local implementation as the code is not available. Note that this method requires the initial location and the orientation and is evaluated only for the $SE(2)$ re-localization task.
- **Conditional Random Field (CRF)** is based on a state-of-the-art map-matching system [28], which computes a reachability graph from a floorplan, uses inertial navigation results to transition between graph nodes, and uses Viterbi algorithm to backtrack and determine location. We modified the system in a few ways to better adapt to our

⁶Movements are governed by architectural structures (e.g., walls and rooms) in University A, where a building blueprint is used as a floorplan. For University B and Office C, transient objects such as chairs and desks play a greater role, but do not show up in blueprints. Therefore, we take the likelihood maps from the synthetic data generation process (See Sect. 5.4) and binary-threshold them as floorplans.

Building	Meth.	Fixed short sequence (100 m)						Full test sequence						run time cpu/gpu (sec) ↓	
		SR(%) at distance ↑			SR(%) at A ↑			SR(%) at distance ↑			SR(%) at A ↑				
		1m	2m	4m	6m	20°	40°	1m	2m	4m	6m	20°	40°		
A	PF	1.8	6.7	11.9	15.4	21.3	32.2	6.5	16.8	22.8	26.2	31.5	42.5	0.6 / 7.7	
	CRF	15.0	32.5	46.3	53.6	61.7	70.5	14.2	31.9	47.0	54.7	53.0	61.0	9.5 / 3.7	
	Ours	16.7	28.9	38.8	44.6	46.8	54.1	23.4	44.8	62.6	69.5	65.6	74.8	0.3 / 0.1	
B	PF	1.0	3.8	7.0	9.0	17.0	27.7	6.4	16.8	28.6	34.0	38.1	51.6	1.8 / 1.4	
	CRF	12.4	33.6	48.7	53.7	62.0	65.7	18.4	49.8	68.6	71.5	71.8	77.2	18.8 / 5.4	
	Ours	47.6	69.3	74.5	77.3	67.9	75.1	49.4	73.1	80.1	82.0	72.7	80.7	1.2 / 0.2	
C	PF	19.7	30.9	46.0	58.6	21.8	38.2	18.3	28.9	43.8	55.2	21.0	38.0	4.3 / 4.2	
	CRF	26.3	36.2	43.7	52.1	31.3	46.3	44.3	60.5	72.1	80.5	44.4	64.9	38.1 / 16.8	
	Ours	69.9	78.1	83.4	87.2	51.8	67.4	72.9	80.5	85.2	89.1	53.4	69.7	2.4 / 0.7	

(a) Inertial Localization

Task	Meth.	Fixed short sequence (100 m)						Full test sequence						run time cpu/gpu (sec) ↓	
		SR(%) at distance ↑			SR(%) at A ↑			SR(%) at distance ↑			SR(%) at A ↑				
		1m	2m	4m	6m	20°	40°	1m	2m	4m	6m	20°	40°		
Reloc \mathbb{R}^2	PF	21.4	40.5	60.3	69.9	48.9	64.1	19.7	39.6	57.0	63.8	48.7	63.0	4.7 / 6.7	
	CRF	33.2	59.7	78.9	87.7	71.7	83.8	31.7	60.3	79.4	86.7	71.7	84.3	21.3 / 8.8	
	Ours	50.9	69.3	77.7	82.0	65.0	74.9	50.8	69.3	78.7	82.6	65.5	76.2	1.3 / 0.3	
Reloc $SE(2)$	PF	22.9	41.5	62.8	73.7	51.5	68.2	15.1	30.8	47.0	54.9	41.8	56.0	2.1 / 6.7	
	LP	9.7	27.1	55.3	70.2	49.2	69.8	4.0	13.2	29.5	40.5	36.9	54.3	7.0 / 2.7	
	CRF	36.5	64.4	82.7	90.6	74.2	86.6	31.9	61.0	79.8	86.9	72.1	85.1	21.4 / 8.8	
	Ours	52.8	71.1	79.4	83.4	66.7	76.6	51.4	70.1	79.6	83.8	66.5	77.4	1.3 / 0.3	

(b) Inertial Re-Localization: average metrics across 3 buildings. (see supplementary for results per building)

Table 2. NILoc achieves competitive accuracy at significantly lower run time. We compare NILoc (ours) with three methods that require a floorplan as input: Particle Filter (PF), Learned Prior (LP) and Conditional Random Fields (CRF). We report success rate (SR) at a given error distance threshold and angle (A) threshold. Run time is the average CPU or GPU time per 1 min of motion sequence. The best and second best results per column are shown in orange and cyan, respectively.

SR(%) at distance →	Localization		Reloc \mathbb{R}^2		Reloc $SE(2)$		Model param.s
	2m	4m	2m	4m	2m	4m	
w/o NILoc (RoNIN [10] only)	-	-	-	-	10.5	25.7	-
w/o loss on the velocity branch	14.8	22.9	16.1	24.3	16.1	24.3	7.5M
w/o velocity compressor	3.6	7.6	6.7	12.3	7.2	12.0	10.2M
w/o TA-location decoder (FC)	5.3	10.4	6.8	13.3	8.6	14.7	211.0M
w/o TA-location decoder (CNN)	39.5	58.3	48.8	68.6	51.1	71.1	10.2M
Ours (velocity branch output)	52.5	72.1	-	-	-	-	10.5M
Ours	44.8	62.6	54.1	70.4	56.0	73.0	10.5M

Table 3. Ablation study. The first row is an inertial navigation algorithm. Next four rows are results after dropping one technical component from our full system. For the 4th and 5th rows, we dropped the translation-aware location decoder and replaced by either a fully connected layer (FC) or a fully convolutional decoder (CNN). The last two rows compare predictions by the velocity branch and the location branch, where the latter is the default prediction reported as our result in other places. The success rate (%) at two distance thresholds (m) on building A are the metrics.

task: 1) use the RoNIN result as the velocity input; 2) increasing the search neighborhood by a factor of 1.5 to handle scale inaccuracies in inertial navigation; and 3) do periodic back-propagation within a fixed window to be comparable with other near-real time baselines.

6.2. Implementation details

We have implemented the proposed system in Pytorch-Lightning [4]. For training, we have used AdamW optimizer [15]. The learning rate has been initialized to 0.0001 with 30 epoch warm-up [7], and reduced by a

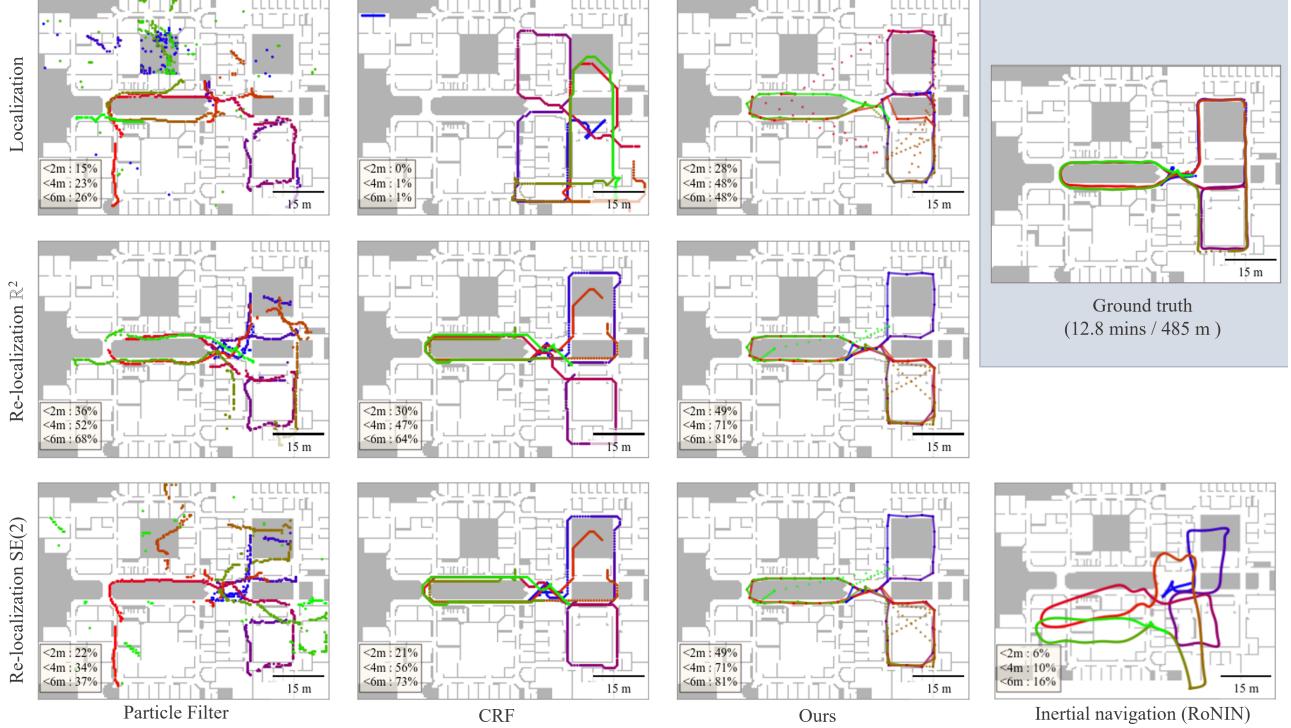


Figure 4. Qualitative visualizations: For one trajectory from building A, we show results by the top three methods (columns) for one localization and two re-localization tasks (rows). Particle filter and CRF require a floorplan in addition to IMU input. The color gradient (blue \rightarrow red \rightarrow green) encodes time. We mark the physical dimension of each sequence and report success rate (%) at distance thresholds 2,4, and 6 meters. See the supplementary for more visualizations.

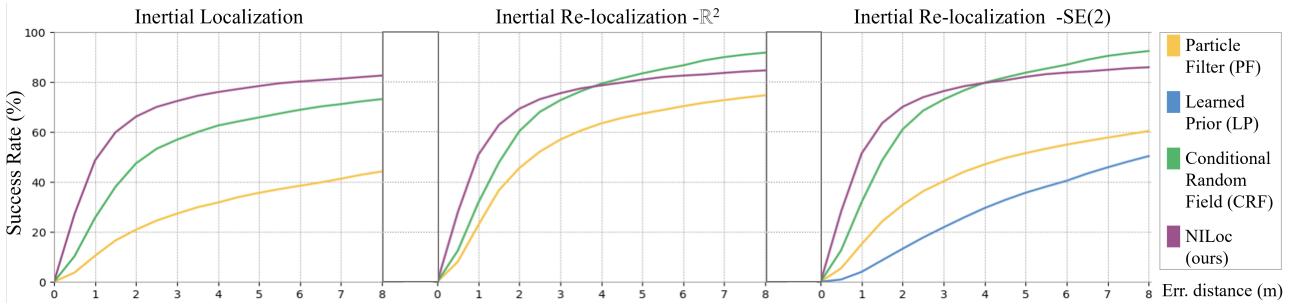


Figure 5. The figure plots the success rate metric for the three tasks, while varying the error distance threshold. All baseline except ours require a floorplan as input in addition to IMU. The average score over the three buildings are reported.

factor of 0.75 after every 10 epochs when the validation loss does not decrease. We use random one sixth of the training trajectories as a hold-out validation. In the main experiments we combine real and synthetic data for training. Specifically, we train on the combined set until convergence (700 epochs) and fine-tune only with real dataset (400 epochs). The total training time is approx. 3 days on GeForce RTX 2080 Ti and 1 day on NVIDIA Tesla V100 GPUs. For RoNIN software, we have downloaded the trained model from the official website [10]. Note that unless otherwise noted, our results indicate the predictions

by the auto-regressive location branch.⁷

For the baseline methods, we use Pytorch [4] and CuPy [19] to implement both on CPU and GPU. For being fair, we use the same floorplan resolution and the same distance-based sampling to extract velocities. A grid search is used to find hyper parameters for all baselines.

⁷For the localization task, we initialize with uniform distribution of particles or location likelihood. For the re-localization task with \mathbb{R}^2 (resp. $SE(2)$), we initialize particles from a Gaussian distribution around ground truth location and uniform distribution for heading (resp. Gaussian distribution around location and heading), or provide ground-truth likelihood map of the first frame (resp. the first two input frames).

6.3. Evaluation

Table 2a is our main result, quantitative evaluations on the localization task for the three buildings separately. NILoc achieves the best results in most entries. The only exception is against CRF for building A for fixed short sequences. Note that CRF uses floorplan information while our input is only IMU. CRF is computationally intensive, 30 times slower than our method, involving even a dynamic programming to effectively search for all possible alignments with a floorplan exhaustively.

Table 2b shows the results on the re-localization task, averaged over the three buildings. All methods consistently improves performance with more initialization information. While NILoc achieves the best results for lower distance thresholds (i.e., often extremely accurate), CRF performs better overall at the sacrifice of its intensive computational expenses and the requirement on the floorplan image.

Figure 5 observes the same results, plotting the distance-based success rate over a range of thresholds (averaged over the three buildings). Particle filter exhibits poor performance, where the major limitation comes from its inability to handle cumulative drifts in the inertial navigation trajectories. Learned prior combines particle filter and neural networks that learn to associate motions with locations. However, they only encode 5 seconds of motion data with LSTM and ConvNet, not long enough to break the ambiguity. NILoc takes roughly a minute of motion data with powerful Transformer based architecture, overcoming the uncertainties. Another observation is that all the baseline methods explicitly integrate velocities to update the location information. NILoc does not bake-in this integration formula and relies completely on learning to relate velocities to locations, which could potentially make our approach more robust against cumulative drifts by inertial navigation.

Figure 4 provides qualitative visualizations of one trajectory from building A. Particle filter and CRF estimate locations at roughly 200Hz, while NILoc is roughly 20Hz. We interpolate NILoc locations in plotting the trajectories, where streaks of points in the figure are interpolation artifacts at discontinuous predictions. As shown by the inertial navigation trajectory at the bottom right, this is not an easy task where the trajectory suffers from significant cumulative drifts. Nonetheless, our system is capable of inferring correct locations for most of the frames.

6.4. Ablation study

Table 3 is an ablation study, assessing the contributions of various technical components in our system. The first row compares against a state-of-art inertial navigation method which ours and all baselines outperform (also see Fig. 4). The next four rows show the distance-based success rate while dropping four components one by one from our

main system. The table shows that it is important to train the network with losses on both branches. The second row is particularly interesting. Both the location and the (first half of the) velocity branches are trained with the loss only at the location branch, whose performance drops significantly. The third row indicates the challenge of high uncertainty in the inertial navigation task. The success rates even drop to a single digit, when the input motion history becomes 10 times less without the TCN based compressor. The last two rows compare the predictions by the velocity branch and the location branch of our full system. The velocity branch does not take previous location likelihoods and cannot solve re-localization tasks. However, for the localization tasks, it outperforms the location branch with a clear margin, while being twice as computationally efficient.

7. Limitations and Future Work

There are two major failure modes in our approach. First, in an open space (e.g., an atrium), human motions tend not to follow patterns, where any location could be an answer. Second, in the presence of symmetries or repetitions, multiple locations would be equally likely. Our method is designed not to use any future frame information after the first input window so that it can be deployed as a real-time system, where predictions may jump abruptly under high uncertainties. Our future work is to exploit body motion signals that are captured in IMU but are currently discarded by inertial navigation and distance-based velocity sampling to overcome the uncertainty. For instance, IMU signals differ when one opens a door, washes hands, or orders a coffee, which provides effective cues in localizing the position. Please refer to the supplementary for more qualitative visualizations (i.e., location trajectories as static images and videos for more samples in more buildings) and quantitative ablation studies (e.g., w/o synthetic data, w/o scheduled sampling, or comparison against TCN as a backbone instead of Transformer). We share our code, models and data to promote further research in the space of inertial localization.

Societal impact: Inertial localization could be a critical component for indoor GPS. A mobile app might become capable of recording location history 24/7 anywhere in indoor spaces, which tend to be more private than outdoors (e.g., inside a house or a rest room). A smartphone developer should understand the impact of giving IMU sensor data and define appropriate access controls for the apps. On the positive side, the inertial location could happen on-device, which would allow a higher degree of privacy control compared to other data modalities.

Acknowledgements: The research is supported by NSERC 8 Discovery Grants, NSERC Discovery Grants Accelerator Supplements, and DND/NSERC Discovery Grant Supplement. We thank Weilian Song, Saghaf Irandoust and Fuyang Zhang for their contribution to dataset.

References

- [1] Naoki Akai and Koichi Ozaki. Gaussian processes for magnetic map-based localization in large-scale indoor environments. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4459–4464. IEEE, 2015. 2
- [2] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018. 4
- [3] Zhenghua Chen, Qingchang Zhu, and Yeng Chai Soh. Smartphone inertial sensor-based indoor localization and tracking with ibeacon corrections. *IEEE Transactions on Industrial Informatics*, 12(4):1540–1549, 2016. 2
- [4] William Falcon and The PyTorch Lightning team. PyTorch Lightning, 3 2019. 6, 7
- [5] Google. Fused Location Provider API. <https://developers.google.com/location-context/fused-location-provider>. 1
- [6] Google. Project tango. <https://get.google.com/tango/>. 3
- [7] Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017. 6
- [8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2
- [9] Sachini Herath, Saghaf Irandoust, Bowen Chen, Yiming Qian, Pyojin Kim, and Yasutaka Furukawa. Fusion-DHL: Wifi, imu, and floorplan fusion for dense history of locations in indoor environments. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5677–5683, 2021. 2
- [10] Sachini Herath, Hang Yan, and Yasutaka Furukawa. RoNIN: Robust neural inertial navigation in the wild: Benchmark, evaluations, & new methods. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3146–3152. IEEE, 2020. 1, 2, 3, 6, 7
- [11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 2, 5
- [12] Yoshihiko Kamiya, Yanlei Gu, and Shunsuke Kamijo. Indoor positioning in large shopping mall with context based map matching. In *2019 IEEE International Conference on Consumer Electronics (ICCE)*, pages 1–6. IEEE, 2019. 2
- [13] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015. 2
- [14] Wenxin Liu, David Caruso, Eddy Ilg, Jing Dong, Anastasios I Mourikis, Kostas Daniilidis, Vijay Kumar, and Jakob Engel. TLIO: Tight learned inertial odometry. *IEEE Robotics and Automation Letters*, 5(4):5653–5660, 2020. 1, 2
- [15] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. 6
- [16] Dennis Melamed. Learnable spatio-temporal map embeddings for deep inertial localization. Master’s thesis, Carnegie Mellon University, Pittsburgh, PA, June 2021. 2, 5
- [17] Tsvetomila Mihaylova and André FT Martins. Scheduled sampling for transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 351–356, 2019. 5
- [18] Qun Niu, Tao He, Ning Liu, Suining He, Xiaonan Luo, and Fan Zhou. Mail: Multi-scale attention-guided indoor localization using geomagnetic sequences. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(2):1–23, 2020. 2
- [19] Ryosuke Okuta, Yuya Unno, Daisuke Nishino, Shohei Hido, and Crissman Loomis. CuPy: A NumPy-Compatible Library for NVIDIA GPU Calculations. In *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)*, 2017. 7
- [20] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, volume 9351 of *LNCS*, pages 234–241. Springer, 2015. 5
- [21] Dimitrios Sikeridis, Bhaskar Prasad Rimal, Ioannis Papapanagiotou, and Michael Devetsikiotis. Unsupervised crowd-assisted learning enabling location-aware facilities. *IEEE Internet of Things Journal*, 5(6):4699–4713, 2018. 2
- [22] Scott Sun, Dennis Melamed, and Kris Kitani. Idol: Inertial deep orientation-estimation and localization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 6128–6137, 2021. 1, 2
- [23] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010. 2
- [24] Hajime Taira, Masatoshi Okutomi, Torsten Sattler, Mircea Cimpoi, Marc Pollefeys, Josef Sivic, Tomas Pajdla, and Akihiko Torii. InLoc: Indoor visual localization with dense matching and view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7199–7209, 2018. 2
- [25] Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002. 2
- [26] Joaquín Torres-Sospedra, David Rambla, Raul Montoliu, Oscar Belmonte, and Joaquín Huerta. Ujiindoorloc-mag: A new database for magnetic field-based localization problems. In *2015 International conference on indoor positioning and indoor navigation (IPIN)*, pages 1–10. IEEE, 2015. 2
- [27] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 2, 4
- [28] Zhuoling Xiao, Hongkai Wen, Andrew Markham, and Niki Trigoni. Lightweight map matching for indoor

- localisation using conditional random fields. In *IPSN-14 Proceedings of the 13th International Symposium on Information Processing in Sensor Networks*, pages 131–142. IEEE, 2014. 5
- [29] Chouchang Yang and Huai-Rong Shao. WiFi-based indoor positioning. *IEEE Communications Magazine*, 53(3):150–157, 2015. 2
- [30] Kwang Moo Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. Lift: Learned invariant feature transform. In *European conference on computer vision*, pages 467–483. Springer, 2016. 2
- [31] Baoding Zhou, Qingquan Li, Qingzhou Mao, Wei Tu, Xing Zhang, and Long Chen. ALIMC: Activity Landmark-Based Indoor Mapping via Crowdsourcing. volume 16, pages 2774–2785. Institute of Electrical and Electronics Engineers Inc., oct 2015. 2