Neural Inertial Localization
Supplementary Material

Sachini Herath[1] David Caruso[2] Chen Liu[2] Yufan Chen[2] Yasutaka Furukawa[1]

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

The supplementary document provides:

- Algorithmic details (Sec. 1);
- More qualitative visualization (Fig. 1 and 2);
- Failure cases (Fig. 3);
- Architecture specification (Tab. 1);
- Quantitative evaluations on re-localization task (Tab. 2);
- More ablation studies (Tab. 3).

Please also refer to our supplementary video, which shows predicted likelihoods and motion trajectories for each method in animtaion.

1. Algorithmic details

We discuss the details of our synthetic data generation process and data augmentation during training, outlined in section 5.4.

1.1. Synthetic data generation

When a floorplan image is not available, we compute the walklable region or a floorplan (F_{map}) by 1) counting the number of times training trajectories pass through at each pixel; 2) clamping the count to be 2 at the maximum; 3) applying Gaussian smoothing ($\sigma = 1.0$ pixels); and 4) binary-thresholding the map with a threshold of 0.5.

We generate a synthetic motion trajectory from the floorplan (F_{map}) by picking start and goal positions randomly, and finding a shortest path between them, while we define a local neighborhood system to be a 11x11 square region.

Next, in order to make the trajectories look realistic, we apply B-spline smoothing then solve an optimization problem so that the trajectories pass through near the center of corridors and passages. Note that when synthetic trajectories are used in training, to minimize the gap between synthetic data and real data, we also apply the B-spline smoothing to the real trajectory before passing the velocity vectors to the velocity branch.

B-spline smoothing: We approximate synthetic trajectory by fitting a b-spline curve [1]. Given a trajectory $\vec{p_t}$ with timestamps $t \in T$, we find a smooth spline approximation of degree 3 with smoothing condition s(=5.0) by "scipy.interpolate.splrep". The b-spline knot vector and control points of the approximated curve are k and $\vec{c}$.

$$B(k, \vec{c}) = F_{Bspline}([\vec{p_t}], T, s) \tag{1}$$

Optimizing with floorplan: Given a b-spline knot vector k and control points $\vec{c}$ of the approximated curve for synthetic data, and a walkable region, we optimize the location of control points, $\vec{c_m}$, using non-linear least squares to minimize the following energy function.

$$\arg\min_{\vec{c_m}} \sum_{t \in T} f_{map}(B(k, \vec{c_m})(t)) + 2.0 \times \|B(k, \vec{c_m})(t) - \vec{p_t}\|$$

where f_{map} is a function of distance from a non-walkable pixel, which is computed by a flood-fill algorithm and smoothed by the Gaussian function ($\sigma = 2.0$). We sample the smoothed spline at constant distance of 1 pixel with a Gaussian noise ($\sigma = 3.0$) to obtain velocity vectors and ground-truth positions.

1.2. Data Augmentation

We further perform the following three data augmentation tricks. First, to ensure that motion directions are not restricted to a discrete set, we randomly rotate the walkable region image by "scipy.ndimage.rotate" function before generating synthetic trajectories by the shortest path algorithm. Second, we add random perturbations to velocity magnitude and angular rate of the input 2D velocity vector sequence by the Gaussian function ($\sigma_{scale} = 0.2$ pixels and $\sigma_{bias} = 0.05$ radians per frame) to mimic scaling errors in inertial navigation and IMU gyroscope bias errors. Third we perform a random in-plane rotation on the input velocity sequence to ensure the learned features are invariant to the unknown starting orientation alignment.

References

[1] Paul Dierckx. Algorithms for smoothing data with periodic and parametric splines. *Computer Graphics and Image Processing*, 20(2):171–184, 1982. 1

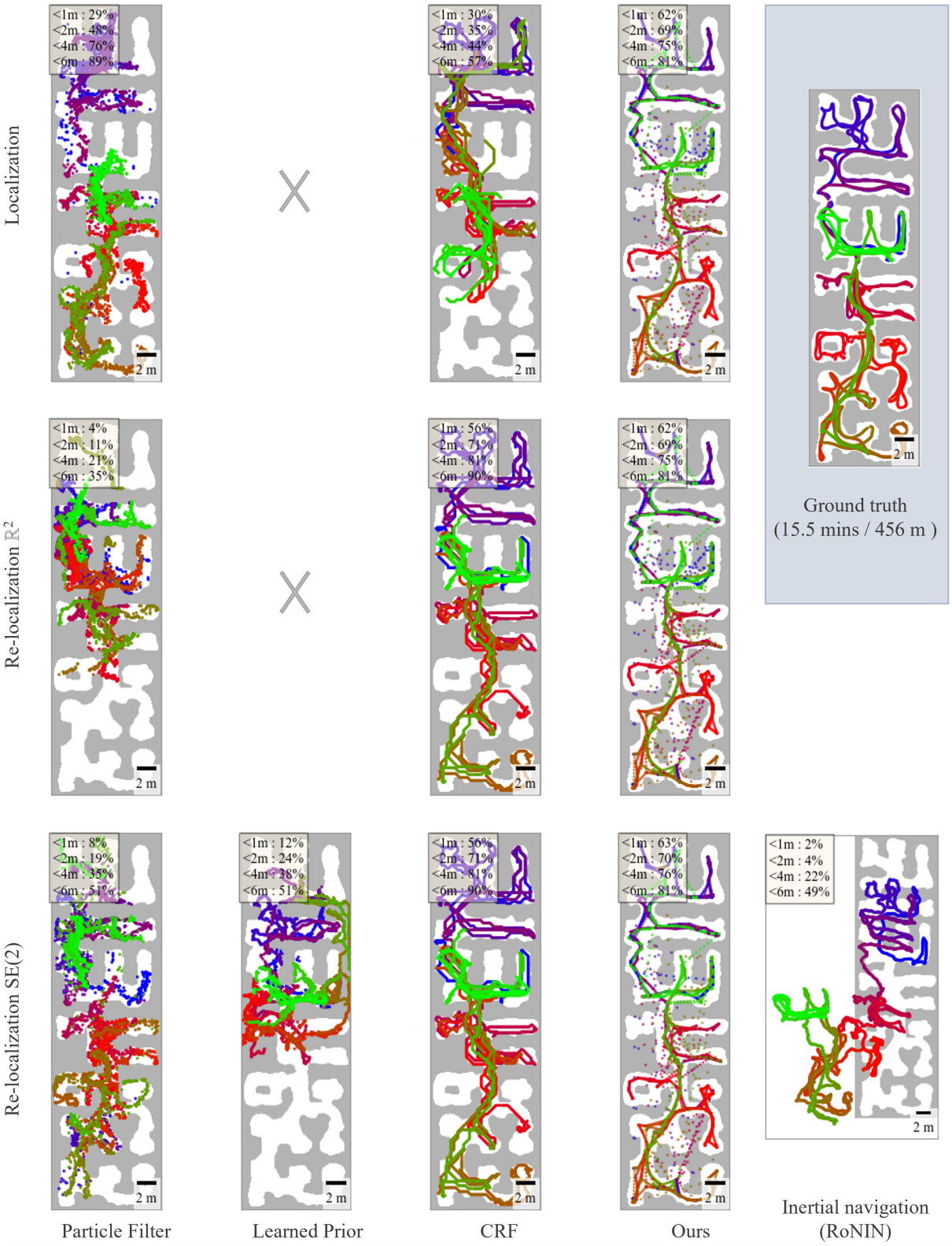


Figure 1. Qualitative visualizations: For one trajectory from Office C, we show results by the four methods (columns) for one localization and two re-localization tasks (rows). Particle filter, learned prior and CRF require a floorplan in addition to IMU input. The color gradient (blue $\rightarrow$ red $\rightarrow$ green) encodes time. We mark the physical dimension of each sequence and report success rate (%) at distance thresholds 1, 2,4, and 6 meters.

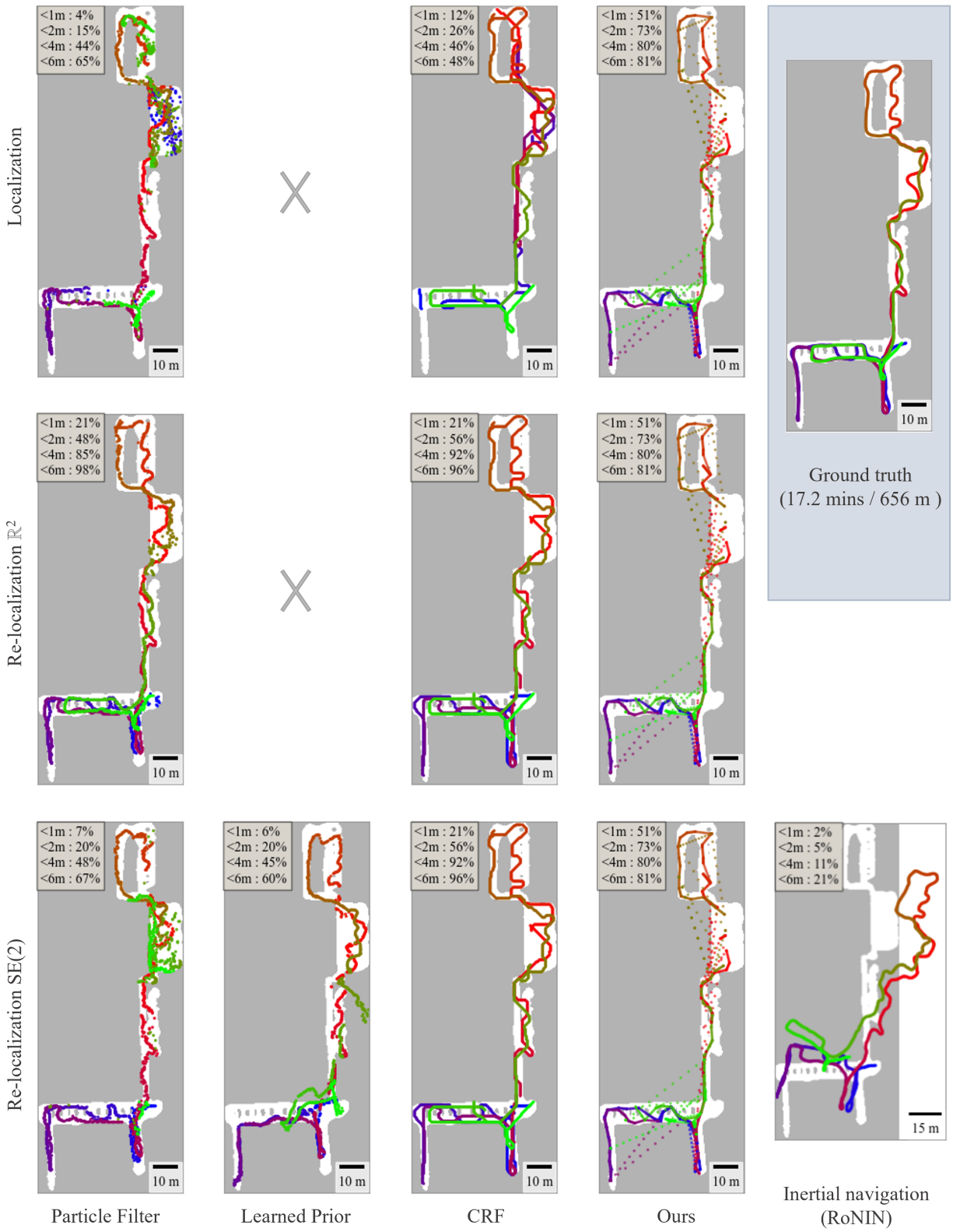
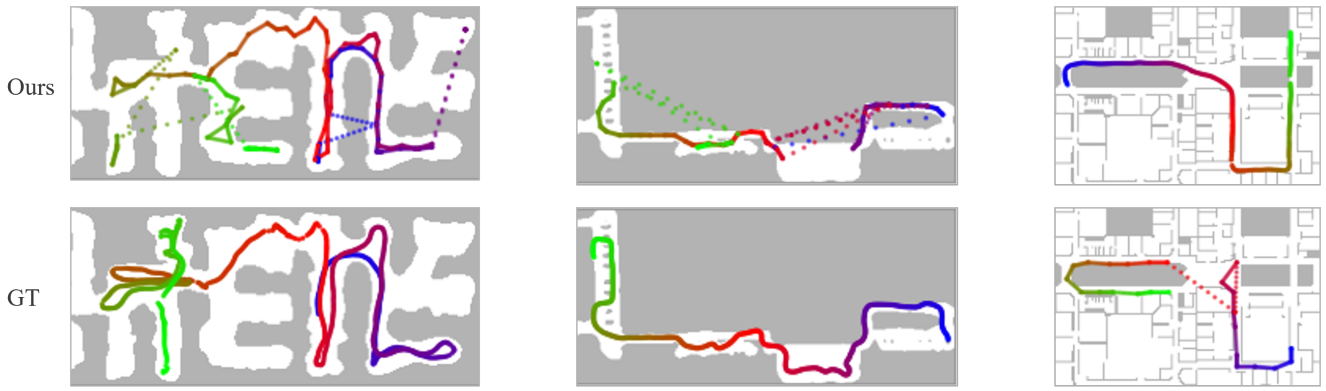


Figure 2. Qualitative visualizations: For one trajectory from building B, we show results by the four methods (columns) for one localization and two re-localization tasks (rows). Particle filter, learned prior and CRF require a floorplan in addition to IMU input. The color gradient (blue → red → green) encodes time. We mark the physical dimension of each sequence and report success rate (%) at distance thresholds 1, 2, 4, and 6 meters.



(a) discontinuities (b) open spaces (c) symmetries

Figure 3. Failure cases: Three common failure modes in our approach are shown with our prediction for localization task and ground-truth (GT) trajectory segments. The color gradient (blue → red → green) encodes time.

Branch	Module	Layers/Operations	Input Shape	Output Shape
Velocity Branch	Velocity Input		[n, 2, 200]	
	TCN Velocity Compressor	TCN	[n, 2, 200]	[n, 288, 200]
		Down-sample	[n, 288, 200]	[n, 288, 20]
		Positional Encoding	[n, 288, 20]	[n, 432, 20]
		Permute	[n, 432, 20]	[20, n, 432]
	Shared Trans. Velocity Encoder	Transformer Encoder	[20, n, 432]	[20, n, 432]
	Transformer Velocity Encoder	Transformer Encoder	[20, n, 432]	[20, n, 432]
	TA-location decoder	Permute	[20, n, 432]	[n×20, 1, 24, 18]
		Transpose-Conv	[n×20, 1, 24, 18]	[n×20, 3, 211, 157]
		TA-Conv	[n×20, 3, 211, 157]	[n×20, 1, 211, 157]
		Permute	[n×20, 1, 211, 157]	[n, 20, 211, 157]
Location Branch	Location input		[n, 20, 211,157]	
	CNN Location Encoder	Permute	[n, 20, 211, 157]	[n×20, 1, 211, 157]
		Conv	[n×20, 1, 211, 157]	[n×20, 1, 24, 18]
		Permute	[n×20, 1, 24, 18]	[20, n, 432]
		Positional Encoding	[20, n, 432]	[20, n, 432]
	Transformer Location Decoder	Transformer Decoder	[20, n, 432], [20, n, 432]	[20, n, 432]
	TA-location decoder	Permute	[20, n, 432]	[n×20, 1, 24, 18]
		Transpose-Conv	[n×20, 1, 24, 18]	[n×20, 3, 211, 157]
		TA-Conv	[n×20, 3, 211, 157]	[n×20, 1, 211, 157]
		Permute	[n×20, 1, 211, 157]	[n, 20, 211, 157]

Table 1. The input/output dimensions of our network for Building A. The tensors are of batch size n, and the location branch is shown with maximum sequence length 20 corresponding to velocity input sequence length of 200 frames. For buildings B and C, velocity input shape remains the same and location input/output shapes change to [n, 20, 144, 368] and [n, 20, 112, 384] resp., and the inner feature dimensions change accordingly.

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	22.3	43.2	59.4	66.3	60.1	71.5	18.6	41.9	62.3	68.2	62.1	71.8	1.4 / 5.8
	CRF	26.2	52.9	76.2	87.8	81.2	92.6	22.6	52.7	73.8	83.6	79.7	90.8	9.2 / 3.7
	Ours	34.4	59.9	74.5	81.2	74.7	81.8	30.0	54.1	70.4	76.5	70.2	77.9	0.3 / 0.1
B	PF	13.9	34.3	59.7	71.2	59.1	73.4	15.4	39.7	58.2	64.1	57.6	71.1	3.8 / 4.4
	CRF	27.1	65.8	88.0	93.2	88.1	93.9	23.7	62.8	87.1	91.2	87.8	93.9	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	27.9	44.0	61.6	72.1	27.6	47.4	25.3	37.1	50.6	59.0	26.3	46.1	9.0 / 13.3
	CRF	46.5	60.4	72.5	82.2	45.9	64.9	48.7	65.4	77.2	85.2	47.6	68.3	36.0 / 17.4
	Ours	70.7	78.7	84.1	87.6	52.4	68.0	73.1	80.8	85.4	89.3	53.6	69.9	2.4 / 0.7

(a) Inertial Re-localization $\mathbb{R}^2$

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	23.6	42.3	60.2	68.7	62.9	75.1	17.2	36.5	54.4	61.0	56.9	67.6	0.6 / 3.1
	LP	5.9	21.0	46.6	59.8	55.0	78.5	4.8	19.8	40.1	52.9	55.1	76.0	4.6 / 0.3
	CRF	27.5	55.0	77.7	88.8	82.2	93.4	23.3	53.5	74.9	84.4	80.9	92.5	9.5 / 3.7
	Ours	36.1	62.0	76.4	82.7	76.7	83.1	31.2	56.0	73.0	79.8	72.9	81.3	0.3 / 0.1
B	PF	14.9	36.1	62.8	75.8	63.3	81.2	9.5	25.3	41.7	48.4	46.8	61.5	1.4 / 4.4
	LP	6.6	24.6	61.0	73.8	62.2	80.4	2.2	9.9	26.9	37.8	48.7	70.6	1.9 / 0.8
	CRF	29.7	71.2	92.3	96.3	91.5	97.4	23.7	64.3	87.1	91.2	87.8	93.9	18.8 / 5.3
	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	30.1	46.1	65.4	76.6	28.4	48.4	18.5	30.5	45.0	55.4	21.8	38.9	4.3 / 12.7
	LP	16.6	35.6	58.4	76.9	30.4	50.4	6.8	15.8	29.3	41.2	18.1	32.4	14.5 / 7.1
	CRF	52.1	67.1	77.9	86.6	49.0	69.0	48.7	65.4	77.2	85.1	47.6	68.8	36.0 / 17.4
	Ours	71.4	79.3	84.4	87.7	52.7	68.7	73.5	81.2	85.8	89.6	53.9	70.2	2.3 / 0.7

(b) Inertial Re-localization $SE(2)$

Table 2. Evaluation per building for re-localization task (Table 2.b of the main paper contains the average metrics across three buildings). 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, per building. 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.

	Velocity branch		Location branch					
	Localization		Localization		Reloc $\mathbb{R}^2$		Reloc SE(2)	
SR(%) at distance →	2m	4m	2m	4m	2m	4m	2m	4m
w/o transformer encoder [LSTM]	30.5	43.8	-	-	-	-	-	-
w/o transformer encoder [TCN]	31.2	43.2	-	-	-	-	-	-
w/o scheduled sampling	25.3	38.1	9.3	15.2	9.3	15.2	9.4	15.6
w/o synthetic data	42.4	64.5	10.4	18.9	14.1	25.0	17.5	29.1
Ours	52.5	72.1	44.8	62.6	54.1	70.4	56.0	73.0

Table 3. Ablation Study: The first two rows are results after replacing transformer encoder module with different neural architectures, in particular LSTM and TCN. Third and fourth rows show the effectiveness of our training and data augmentation processes resp. The success rate (%) at two distance thresholds (m) on building A are the metrics.