CTIN: Robust Contextual Transformer Network for Inertial Navigation

Bingbing Rao[1], **Ehsan Kazemi**[1, 2], **Yifan Ding**[1], **Devu M. Shila**[2], **Frank M. Tucker**[3], **Liqiang Wang**[1]

[1] Department of Computer Science, University of Central Florida, Orlando, FL, USA
[2] Unknot.id Inc., Orlando, FL, USA
[3] U.S. Army CCDC SC, Orlando, FL, USA

Abstract

Recently, data-driven inertial navigation approaches have demonstrated their capability of using well-trained neural networks to obtain accurate position estimates from inertial measurement units (IMUs) measurements. In this paper, we propose a novel robust **C**ontextual **T**ransformer-based network for **I**nertial **N**avigation (CTIN) to accurately predict velocity and trajectory. To this end, we first design a ResNet-based encoder enhanced by local and global multi-head self-attention to capture spatial contextual information from IMU measurements. Then we fuse these spatial representations with temporal knowledge by leveraging multi-head attention in the Transformer decoder. Finally, multi-task learning with uncertainty reduction is leveraged to improve learning efficiency and prediction accuracy of velocity and trajectory. Through extensive experiments over a wide range of inertial datasets (*e.g.,* RIDI, OxIOD, RoNIN, IDOL, and our own), CTIN is very robust and outperforms state-of-the-art models.

Introduction

Inertial navigation is a never-ending endeavor to estimate the states (*i.e.,* position and orientation) of a moving subject (*e.g.,* pedestrian) by using only IMUs attached to it. An IMU sensor, often a combination of accelerometers and gyroscopes, plays a significant role in a wide range of applications from mobile devices to autonomous systems because of its superior energy efficiency, mobility, and flexibility (Lymberopoulos et al. 2015). Nevertheless, the conventional Newtonian-based inertial navigation methods reveal not only poor performance, but also require unrealistic constraints that are incompatible with everyday usage scenarios. For example, strap-down inertial navigation systems (SINS) may obtain erroneous sensor positions by performing double integration of IMU measurements, duo to exponential error propagation through integration (Titterton, Weston, and Weston 2004). Step-based pedestrian dead reckoning (PDR) approaches can reduce this accumulated error by leveraging the prior knowledge of human walking motion to predict trajectories (Tian et al. 2015). However, an IMU must be attached to a foot in the zero-velocity update (Foxlin 2005) or a subject must walk forward so that the motion direction is constant in the body frame (Brajdic and Harle 2013). In addition, inertial sensors are often combined with additional sensors and models using Extended Kalman Filter (Bloesch et al. 2015) to provide more accurate estimations, where the typical sensors include WiFi (Ahmetovic et al. 2016), Bluetooth (Li, Guo, and Li 2017), LiDAR (Zhang and Singh 2014), or camera sensors (Leutenegger et al. 2015). Nonetheless, these combinations with additional sensors are posing new challenges about instrument installations, energy efficiency, and data privacy. For instance, Visual-Inertial Odometry (VIO) substantially depends on environmental factors such as lighting conditions, signal quality, blurring effects (Usenko et al. 2016).

Recently, a growing number of data-driven approaches such as IONet (Chen et al. 2018), RoNIN (Herath, Yan, and Furukawa 2020), and IDOL (Sun, Melamed, and Kitani 2021) have demonstrated their capability of using well-trained neural networks to obtain accurate estimates from IMU measurements with competitive performance over the aforementioned methods. However, grand challenges still exist when applying neural network techniques to IMU measurements: 1) most existing data-driven approaches leverage sequence-based models (*e.g.,* LSTM (Hochreiter and Schmidhuber 1997)) to learn temporal correlations but fail to capture spatial relationships between multivariate time-series. 2) There is few research work to explore rich *contextual* information among IMU measurements in dimensions of spatial and temporal for inertial feature representation. 3) Usually, uncertainties of IMU measurements and model output are assumed to be a fixed covariance matrix in these pure and black-box neural inertial models, which brings significant inaccuracy and much less robustness because they can fluctuate dramatically and unexpectedly in nature.

In response to the observations and concerns raised above, a novel robust contextual Transformer network is proposed to regress velocity and predict trajectory from IMU measurements. Particularly, CTIN extends the ideas of ResNet-18 (He et al. 2016) and Transformer (Vaswani et al. 2017) to exploit spatial and longer temporal information among IMU observations and then uses the attention technique to fuse this information for inertial navigation. The major contributions of this paper are summarized as follows:

- Extending ResNet-18 with attention mechanisms is to explore and encode spatial information of IMU samples.
- A novel self-attention mechanism is proposed to extract

contextual features of IMU measurements.

- Multi-Task learning using novel loss functions is to improve learning efficiency and reduce models' uncertainty.
- Comprehensive qualitative and quantitative comparisons with the existing baselines indicate that CTIN outperforms state-of-the-art models.
- A new IMU dataset with ground-truth trajectories under natural human motions is provided for future reference.
- To the best of our knowledge, CTIN is the first Transformer-based model for inertial navigation.

Background

IMU models

Technically, 3D angular velocity (ω) and 3D acceleration (α) provided by IMUs are subjected to bias and noise based on some sensor properties, as shown in Equation 1 & 2:

$$\omega_t = r_t^\omega + b_t^\omega + n_t^\omega \tag{1}$$

$$\alpha_t = r_t^\alpha + b_t^\alpha + n_t^\alpha \tag{2}$$

where r_t^ω and r_t^α are real sensor values measured by the gyroscope and accelerometer at timestamp t, respectively; b_t^ω and b_t^α are time-varying bias; n_t^ω and n_t^α are noise values, which usually follow a zero-mean gaussian distribution.

Inertial Tracking

According to Newtonian mechanics (Kok, Hol, and Schön 2017), states (*i.e.,* position and orientation) of a moving subject (*e.g.,* pedestrian) can be estimated from a history of IMU measurements, as shown in Equation 3:

$$R_b^n(t) = R_b^n(t-1) \otimes \Omega(t) \tag{3a}$$

$$\Omega(t) = exp(\frac{dt}{2}\omega(t-1)) \tag{3b}$$

$$v^n(t) = v^n(t-1) + \Delta(t) \tag{3c}$$

$$\Delta(t) = (R_b^n(t-1) \odot \alpha(t-1) - g^n)dt \tag{3d}$$

$$P^n(t) = P^n(t-1) + v^n(t-1)dt \tag{3e}$$

Here, the orientation $R_b^n(t)$ at timestamp t is updated with a relative orientation ($\Omega(t)$) between two discrete instants t and $t-1$ according to Equation 3a & 3b, where $\omega(t-1)$ measures proper angular velocity of an object at timestamp $(t-1)$ in the body frame (denoted by b) with respect to the navigation frame (denoted by n). R_b^n can be used to rotate a measurement $x \in [\omega, \alpha]$ from the body frame b to the navigation frame n, which is denoted by an expression $R_b^n \odot x = R_b^n \otimes x \otimes (R_b^n)^T$ where $\otimes$ is a hamilton product between two quaternions. The navigation frame in our case is defined such that Z axis is aligned with earth's gravity g^n and the other two axes are determined according to the initial orientation of the body frame. In Equation 3c & 3d, velocity vector $v^n(t)$ is updated with its temporal difference $\Delta(t)$, which is obtained by rotating $\alpha(t-1)$ to the navigation frame using $R_b^w(t-1)$ and discarding the contribution of gravity forces g^n. Finally, positions $P^n(t)$ are obtained by integrating velocity in Equation 3e. Therefore, given current IMU measurements (*i.e.,* α, ω), the new system states (*i.e.,* P^n, v^n and R_b^n) can be obtained from the previous states using a function of f in Equation 4, where f represents transformations in Equation 3.

$$[P^n, v^n, R_b^n]_t = f([P^n, v^n, R_b^n]_{t-1}, [\alpha, \omega]_t) \tag{4}$$

Drawback and Solution: However, using IMUs for localization results in significant drift due to that the bias and noise intrinsic to the gyroscope and accelerometer sensing can explode quickly in the double integration process. Using pure data-driven models with IMU measurements for Inertial Navigation has shown promising results in pedestrian dead-reckoning systems. To tackle the problems of error propagation in Equation 4, we break the cycle of continuous integration and segment inertial measurements into independent windows, then leverage a sequence-to-sequence neural network architecture (Sutskever, Vinyals, and Le 2014; Bahdanau, Cho, and Bengio 2015; Wu et al. 2016; Vaswani et al. 2017) to predict velocities and positions from an input window m of IMU measurements, as shown in Equation 5.

$$[P^n, v^n]^{1:m} = \mathcal{F}_\theta(P_0^n, v_0^n, [R_b^n, \alpha, \omega]^{1:m}) \tag{5}$$

where $\mathcal{F}_\theta$ represents a latent neural system that learns the transformation from IMU samples to predict positions and velocities, where P_0^n, v_0^n are initial states.

Attention Mechanism

Attention can be considered as a query procedure that maps a query Q for a set of key-value pairs (K, V) to an output (Vaswani et al. 2017; Han et al. 2020), which is denoted by $ATT(Q, K, V) = \gamma(Q, K) \times V$. Typically, the output is computed as a sum of weighted values (V), where the weights $\gamma(Q, K)$ are computed according to a compatibility function of Q and K. There are two kinds of γ used in this paper (Bahdanau, Cho, and Bengio 2015; Wang et al. 2018): (1) we perform a *dot product* between Q and K, divides each resulting element by $\sqrt{d}$, and applies a *softmax* function to obtain the weights: $\gamma(Q, K) = softmax(\frac{QK^T}{\sqrt{d}})$ where d is the dimension size of vectors Q, K and V. (2) Inspired by Relation Networks (Santoro et al. 2017), we investigate a form of *concatenation*: $\gamma(Q, K) = ReLU(W_\gamma[Q, K])$, where $[\cdot, \cdot]$ denotes concatenation and W_γ is a weight vector that projects the concatenated vector to a scalar. *Self-attention* networks compute a representation of an input sequence by applying attention to each pair of tokens from the sequence, regardless of their distance (Vaswani et al. 2017). Technically, given IMU samples $X \in \mathbb{R}^{m \times d}$, we can perform the following transformation on X directly to obtain Q, K and V: $Q, K, V = XW_Q, XW_K, XW_V$, where $\{W_Q, W_K, W_V\} \in R^{d \times d}$ are trainable parameters. Usually, these intermediate vectors are split into different representation subspaces at different positions (*i.e.,* $h = 8, d_k = \frac{d}{h}$), *e.g.,* $K = [K^1, \ldots, K^h]$ with $K^i \in \mathbb{R}^{m \times d_k}$. For a subspace, the attention output is calculated by $head_i = ATT(Q^i, K^i, V^i)$. The final output representation is the concatenation of outputs generated by multiple attention heads: $MultiHead(Q, K, V) = [head_i, \ldots, head_h]$.

Our Approach

System Overall

The Attention-based architecture for inertial navigation is shown in Figure 1 and its workflow is depicted as follows:

Data Preparation. Initially, an IMU sample is the concatenation of data from gyroscope and accelerometer. To exploit temporal characteristics of IMU samples, we leverage

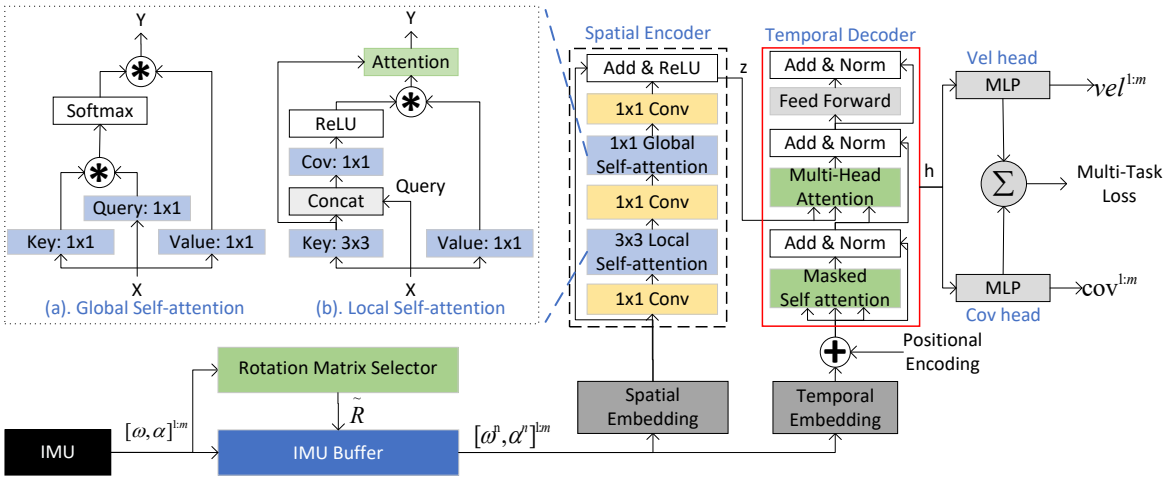


Figure 1: Overall workflow of the proposed contextual transformer model for inertial navigation.

a sliding window with size m to prepare datasets at timestamp t, denoted by $X_t^{1:m} = [x_{t-m+1}, \ldots, x_t]$. Similarly, we adopt this rolling mechanism with the same window size to build the ground truth of velocities: $gt_{vel}^{1:m}$. Usually, IMU samples in each window are rotated from the body frame (*i.e.,* ω^b, α^b) to the navigation frame (*i.e.,* ω^n, α^n) using provided orientations. *Rotation Matrix Selector* is designed to select sources of orientation for training and testing automatically. Typically, we use the device orientation estimated from IMU for testing.

Embedding. We need to compute feature representations for IMU samples before feeding them into encoder and decoder. *Spatial Embedding* uses a 1D convolutional neural network followed by batch normalization and linear layers to learn spatial representations; *Temporal Embedding* adopts a 1-layer bidirectional LSTM model to exploit temporal information, and then adds positional encoding provided by a trainable neural network.

Spatial Encoder. The encoder comprises a stack of N identical layers, which maps an input sequence of $X_t^{1:m}$ to a sequence of continuous representations $z = (z_1, \ldots, z_m)$. To capture spatial knowledge of IMU samples at each timestamp, we strengthen the functionality of the core bottleneck block in ResNet-18 (He et al. 2016) by replacing spatial convolution with a local self-attention layer and inserting a global self-attention module before the last 1×1 downsampling convolution (cf. in Section). All other structures, including the number of layers and spatial downsampling, are preserved. The modified bottleneck layer is repeated multiple times to form *Spatial Encoder*, with the output of one block being the input of the next one.

Temporal Decoder. The decoder also comprises a stack of N identical layers. Within each layer, we first perform a masked self-attention sub-layer to extract dependencies in the temporal dimension. The masking emphasizes a fact that the output at timestamp t can depend only on IMU samples at timestamp less than t. Next, we conduct a multi-head attention sub-layer over the output of the encoder stack to fuse spatial and temporal information into a single vector representation and then pass through a position-wise fully connected feed-forward sub-layer. We also employ residual connections around each of the sub-layers, followed by layer normalization.

Velocity and Covariance. Finally, two MLP-based branch heads regress 2D velocity ($vel_t^{1:m}$) and the corresponding covariance matrix ($cov_t^{1:m}$) using the input of h, respectively. Position can be obtained by the integration of velocity. The model of the covariance, denoted by $\mathbf{\Sigma} : x \rightarrow \mathbb{R}^{2\times 2}$ where x is a system state, can describe the distribution difference between ground-truth velocity and the corresponding predictions of them during training. Given that, the probability of a velocity y_v considering current system state x can be approximated by a multivariate Gaussian distribution (Russell and Reale 2021):

$$p_c(y_v|x) = \frac{1}{\sqrt{(2\pi)^2|\mathbf{\Sigma}(x)|}} \times exp(-\frac{1}{2}(y_v - \mathcal{F}_\theta(x))^T \mathbf{\Sigma}(\boldsymbol{x})^{-1}(y_v - \mathcal{F}_\theta(x))) \quad (6)$$

It is worthwhile to mention that we also leverage multi-task learning with uncertainty reduction to accomplish the desired performance (See details in Section).

Attention In Inertial Navigation

In this paper, the encoder and decoder rely entirely on attention mechanism with different settings for embedding matrix $\{W_Q, W_K, W_V\}$ and γ to explore spatial and temporal knowledge from IMU samples.

Global self-attention in Encoder. It triggers the feature interactions across different spatial locations, as shown in Figure 1(a). Technically, we first transform X into Q, K, and V using three separated 1D 1×1 convolutions, respectively. After that, we obtain the global attention matrix (*i.e.,* $\gamma(Q, K)$) between K and Q using a *Dot Product* version of γ. Finally, the final output Y is computed by $\gamma(Q, K) \times V$. In addition, we also adopt multi-head attention to jointly summarize information from different sub-space representations at different spatial positions.

Dataset	Year	IMU Carrier	Sample Frequency	N^{o} of Subjects	N^{o} of Sequences	Ground Truth	Motion Context
RIDI	2017	Lenovo Phab2 Pro	200 Hz	10	98	Google Tango phone	Four attachments: leg pocket, bag, hand, body
OxIOD	2018	iPhone 5/6, 7 Plus, Nexus 5	100 Hz	5	158	Vicon	Four attachments: handheld, pocket, handbag, trolley
RoNIN	2019	Galaxy S9, Pixel 2 XL	200 Hz	100	276	Asus Zenfone AR	Attaching devices naturally
IDOL	2020	iPhone 8	100 Hz	15	84	Kaarta Stencil	Attaching devices naturally
CTIN	2021	Samsung Note, Galaxy	200 Hz	5	100	Google ARCore	Attaching devices naturally

Table 1: Description of public datasets used for evaluation of navigation models.

Local self-attention in Encoder. Although performing a global self-attention over the whole feature map can achieve competitive performance, it not only scales poorly but also misses contextual information among neighbor keys. Because it treats queries and keys as a group of isolated pairs and learns their pairwise relations independently without exploring the rich contexts between them. To alleviate this issue, a body of research work (Hu et al. 2019; Ramachandran et al. 2019; Zhao, Jia, and Koltun 2020; Li et al. 2021; Yao et al. 2022) employs self-attention within the local region (*i.e.,* 3×3 grid) to boost self-attention learning efficiently, and strengthen the representative capacity of the output aggregated feature map. In this paper, we follow up this track and design a novel local self-attention for inertial navigation, as shown in Figure 1(b). In particular, we first employ 3×3 group convolution over all the neighbor keys within a grid of 3×3 to extract local contextual representations for each key, denoted by $C_1 = XW_{K,3\times3}$. After that, the attention matrix (*i.e.,* $\gamma(Q, C_1)$) is achieved through a *concatenation* version of γ in which W_γ is a 1×1 convolution and Q is defined as X. Next, we calculate the attended feature map C_2 by $\gamma(Q, C_1) \times V$, which captures the global contextual interactions among all IMU samples. The final output Y is fused by an attention mechanism between local context C_1 and global context C_2.

Multi-head attention in Decoder. We inherit settings from vanilla Transformer Decoder for attention mechanisms (Vaswani et al. 2017). In other words, we take three separated linear layers to generate Q, K and V from X, respectively, and leverage a pairwise function of *Dot product* to calculate attention matrix (*i.e.,* $\gamma(Q, K)$). Finally, the final output Y is computed by $\gamma(Q, K) \times V$.

Jointly Learning Velocity and Covariance

We leverage multi-task learning with uncertainty reduction to improve learning efficiency and prediction accuracy of the two regression tasks: prediction of 2D velocity and its covariance. Inspired by (Kendall, Gal, and Cipolla 2018; Liu et al. 2020; Yao et al. 2021; Yang et al. 2021), we derive a multi-task loss function by maximizing the Gaussian likelihood with uncertainty (Kendall and Gal 2017). First, we define our likelihood as a Gaussian with mean given by the model output as $p_u(y|\mathcal{F}_\theta(x)) = \mathcal{N}(\mathcal{F}_\theta(x), \delta^2)$, where δ is an observation noise scalar. Next, we derive the model's minimization objective as a Negative Log-Likelihood (NLL) of two model outputs y_v (velocity) and y_c (covariance):

$$
\begin{aligned}
&\mathcal{L}(\mathcal{F}_\theta, \delta_v, \delta_c) \\
&= -\log(p_u(y_v, y_c|\mathcal{F}_\theta(x))) \\
&= -\log(p_u(y_v|\mathcal{F}_\theta(x)) \times p_u(y_c|\mathcal{F}_\theta(x))) \\
&= -(\log(p_u(y_v|\mathcal{F}_\theta(x))) + \log(p_u(y_c|\mathcal{F}_\theta(x))) \\
&= -(\log(\mathcal{N}(y_v; \mathcal{F}_\theta(x), \delta_v^2)) + \log(\mathcal{N}(y_c; \mathcal{F}_\theta(x), \delta_c^2))) \\
&\propto \underbrace{\frac{\parallel y_v - \mathcal{F}_\theta(x) \parallel^2}{2\delta_v^2} + \log \delta_v}_{\text{Velocity}} + \underbrace{\frac{\parallel y_c - \mathcal{F}_\theta(x) \parallel^2}{2\delta_c^2} + \log \delta_c}_{\text{Covariance}} \\
&= \frac{1}{2\delta_v^2}\mathcal{L}_v + \frac{1}{2\delta_c^2}\mathcal{L}_c + \log \delta_v \delta_c
\end{aligned}
\tag{7}
$$

where δ_v and δ_c are observation noises for velocity and covariance, respectively. Their loss functions are denoted by $\mathcal{L}_v$ and $\mathcal{L}_c$, and depicted as follows:

Integral Velocity Loss (IVL, $\mathcal{L}_v$). Instead of performing mean square error (MSE) between predicted velocity ($\hat{v}$) and the ground-truth value (v), we first integrate predicted positions from $\hat{v}$ (cf. Equation 3e), and then define a L2 norm against the ground-truth positional difference within same segment of IMU samples, denoted by $\mathcal{L}_v^p$. In addition, we calculate cumulative error between $\hat{v}$ and v, denoted by $\mathcal{L}_v^e$. Finally, $\mathcal{L}_v$ is defined as $\mathcal{L}_v^p + \mathcal{L}_v^e$.

Covariance NLL Loss (CNL, $\mathcal{L}_c$). According to the covariance matrix in Equation 6, We define the Maximum Likelihood loss as the NLL of the velocity with consideration of its corresponding covariance $\mathbf{\Sigma}$:

$$
\begin{aligned}
\mathcal{L}_c &= -\log(p_c(y_v|x)) \\
&= \frac{1}{2}(y_v - f(x))^T \mathbf{\Sigma}(\boldsymbol{x})^{-1}(y_v - f(x)) + \frac{1}{2}\ln|\mathbf{\Sigma}(\boldsymbol{x})| \\
&= \frac{1}{2}\parallel y_v - f(x) \parallel^2_{\mathbf{\Sigma}(\boldsymbol{x})} + \frac{1}{2}\ln|\mathbf{\Sigma}(\boldsymbol{x})|
\end{aligned}
\tag{8}
$$

There is a rich body of research work to propose various covariance parametrizations for neural network uncertainty estimation (Liu et al. 2020; Russell and Reale 2021). In this study, we simply define the variances along the diagonal, which are parametrized by two coefficients of a velocity.

Experiments

We evaluate CTIN on five datasets against four representative prior research works. CTIN was implemented in Pytorch 1.7.1 (Paszke et al. 2019) and trained using Adam optimizer (Kingma and Ba 2014). During training, early stopping with 30 patience (Prechelt 1998; Wang et al. 2020) is leveraged to avoid overfitting according to model performance on the

Dataset	Test Subject	Metric	Performance (meter)							Perf. Improvement		
			SINS	PDR	RIDI	RoNIN			CTIN	CTIN improvement over RoNIN		
						R-LSTM	R-ResNet	R-TCN		R-LSTM	R-ResNet	R-TCN
RIDI	Seen	ATE	6.34	22.76	8.18	2.55	2.33	3.25	**1.39**	45.36%	40.10%	57.13%
		T-RTE	8.13	24.89	9.34	2.34	2.36	2.64	**1.99**	15.00%	15.78%	24.80%
		D-RTE	0.52	1.39	0.97	0.16	0.16	0.17	**0.11**	32.47%	32.26%	35.91%
	Unseen	ATE	4.62	20.56	8.18	2.78	1.97	2.06	**1.86**	33.07%	5.40%	9.68%
		T-RTE	4.58	31.17	10.51	2.95	2.47	**2.43**	2.49	15.66%	-0.70%	-2.36%
		D-RTE	0.36	1.19	1.09	0.15	0.14	0.14	**0.11**	28.00%	21.22%	22.72%
OxIOD	Seen	ATE	15.36	9.78	3.78	3.87	2.40	3.33	**2.32**	40.10%	3.52%	30.27%
		T-RTE	11.02	8.51	3.99	1.56	1.83	1.49	**0.62**	60.40%	66.27%	58.67%
		D-RTE	0.96	1.16	2.30	0.20	0.56	0.19	**0.07**	61.94%	86.67%	61.21%
	Unseen	ATE	13.90	17.72	7.16	5.22	3.51	6.16	**3.34**	35.90%	4.61%	45.69%
		T-RTE	10.51	17.21	7.65	2.65	2.51	2.61	**1.33**	50.00%	47.18%	49.15%
		D-RTE	0.89	1.10	2.62	0.29	0.49	0.24	0.13	55.57%	73.45%	45.48%
RoNIN	Seen	ATE	7.89	26.64	16.82	5.11	**3.99**	6.18	4.62	9.49%	-15.81%	25.23%
		T-RTE	5.30	23.82	19.50	3.05	2.83	3.27	**2.81**	7.70%	0.69%	13.91%
		D-RTE	0.42	0.98	4.99	0.22	0.19	0.20	**0.18**	18.94%	2.75%	10.15%
	Unseen	ATE	7.62	23.49	15.75	8.73	5.76	7.49	**5.61**	35.77%	2.60%	25.11%
		T-RTE	5.12	23.07	19.13	4.87	4.50	4.70	**4.48**	8.04%	0.42%	4.61%
		D-RTE	0.43	1.00	5.37	0.29	**0.25**	0.26	**0.25**	12.63%	0%	4.83%
IDOL	Seen	ATE	21.54	18.44	9.79	4.57	4.44	4.68	**2.90**	36.49%	34.63%	37.98%
		T-RTE	14.93	14.53	7.97	1.72	1.58	1.77	**1.35**	21.47%	14.54%	23.46%
		D-RTE	1.07	1.14	0.97	0.19	0.26	0.18	**0.13**	28.39%	48.21%	25.12%
	Unseen	ATE	20.34	16.83	9.54	5.60	3.81	5.89	**3.69**	34.19%	3.28%	37.40%
		T-RTE	18.48	15.67	9.07	1.99	1.67	2.21	**1.65**	16.73%	1.02%	25.30%
		D-RTE	1.36	1.31	1.04	0.20	0.22	0.20	**0.15**	25.36%	30.14%	25.52%
CTIN	Seen	ATE	5.63	12.05	4.88	2.22	2.39	2.02	**1.28**	42.25%	46.45%	36.68%
		T-RTE	5.34	16.39	4.21	2.10	2.01	1.73	**1.29**	38.54%	35.87%	25.55%
		D-RTE	0.50	0.79	0.18	0.11	0.16	0.11	**0.08**	28.91%	50.56%	24.61%

Table 2: Overall Trajectory Prediction Accuracy. The best result is shown in bold font.

validation dataset. To be consistent with the experimental settings of baselines, we conduct both training and testing on NVIDIA RTX 2080Ti GPU.

Dataset and Baseline

As shown in Table 1, all selected datasets with rich motion contexts (*e.g.,* handheld, pocket) are collected by multiple subjects using two devices: one is to collect IMU measurements and the other provides ground truth (*i.e.,* position). All datasets are split into training, validation, and testing datasets in a ratio of 8:1:1. For testing datasets except in CTIN, there are two sub-sets: one for subjects that are also included in the training and validation sets, the other for unseen subjects. The selected baseline models are listed below:

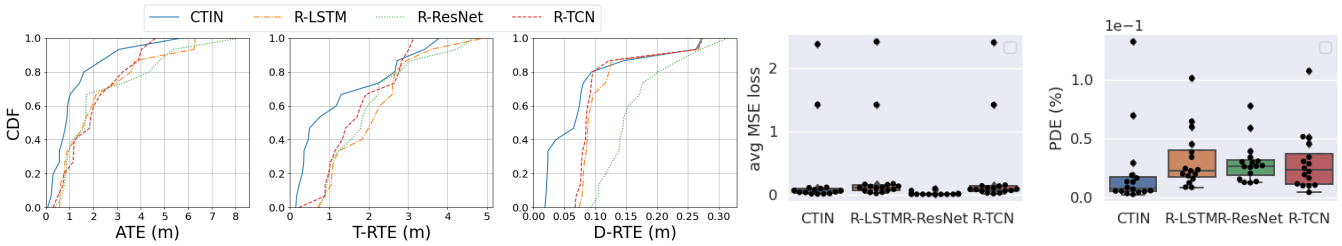
- *Strap-down Inertial Navigation System (SINS):* The subject's position can be obtained from double integration of linear accelerations (with earth's gravity subtracted). To this end, we need to rotate the accelerations from the body frame to the navigation frame using device orientations and perform an integral operation on the rotated accelerations twice to get positions (Savage 1998).
- *Pedestrian Dead Reckoning (PDR):* We leverage Adaptiv[1] to detect foot-steps and update positions per step along the device heading direction. We assume a stride length of 0.67m/step.
- *Robust IMU Double Integration (RIDI):* We use the original implementation (Yan, Shan, and Furukawa 2018) to train a separate model for each device attachment in RIDI and OxIOD datasets. For the rest of the datasets, we train a unified model for each dataset separately, since attachments during data acquisition in these datasets are mixed.
- *Robust Neural Inertial Navigation (RoNIN):* We use the original implementation (Herath, Yan, and Furukawa 2020) to evaluate all three RoNIN variants (*i.e.,* R-LSTM, R-ResNet, and R-TCN) on all datasets.

[1]An Adaptive Jerk Pace Buffer Step Detection Algorithm

Evaluation Metrics

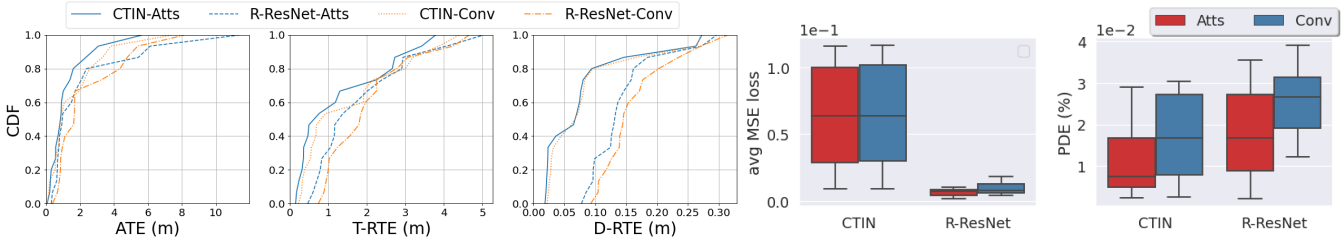
Usually, positions in trajectory can be calculated by performing integration of velocity predicted by CTIN. The major metric used to evaluate the accuracy of positioning is a Root Mean Squared Error (RMSE) with various definitions of estimation error: $RMSE = \sqrt{\frac{1}{m}\sum_{t=1}^{m} \parallel E_t(x_t, \tilde{x}_t) \parallel}$, where m means the number of data points; $E_t(x_t, \tilde{x}_t)$ represents an estimation error between a position (*i.e.,*x_t) in the ground truth trajectory at timestamp t and its corresponding one (*i.e.,* $\tilde{x}_t$) in the predicted path. In this study, we define the following metrics (Sturm et al. 2011):

- **Absolute Trajectory Error (ATE)** is the RMSE of estimation error: $E_t = x_t - \tilde{x}_t$. The metric shows a global consistency between the trajectories and the error is increasing by the path length.



(a) (b)

Figure 2: Performance Comparison of CTIN and RoNIN variant models on CTIN dataset



(a) (b)

Figure 3: The effectiveness of proposed attention layers on CTIN dataset. "*-atts" means CTIN or R-ResNet models with attention functionalities; "*-Conv" represents the models using a conventional spatial convolution instead.

- **Time-Normalized Relative Traj. Error (T-RTE)** is the RMSE of average errors over a time-interval window span (*i.e.,* $t_i = 60$ seconds in our case). The estimation error is defined formally as $E_t = (x_{t+t_i} - x_t) - (\tilde{x}_{t+t_i} - \tilde{x}_t)$. This metric measures the local consistency of estimated and ground truth path.
- **Distance Normalized Relative Traj. Error (D-RTE)** is the RMSE across all corresponding windows when a subject travels a certain distance d, like d is set to 1 meter in our case. The estimation error is given by $E_t = (x_{t+t_d} - x_t) - (\hat{x}_{t+t_d} - \hat{x}_t)$ where t_d is the time interval needed to traverse a distance of d.
- **Position Drift Error (PDE)** measures final position (at timestamp m) drift over the total distance traveled (*i.e.,* traj._len): $(\parallel x_m - \hat{x}_m \parallel) \ / \ \text{traj.\_len}$

Overall Performance

Table 2 shows experimental trajectory errors across entire test datasets. It demonstrates that CTIN can achieve the best results on most datasets in terms of ATE, T-RTE, and D-RTE metrics, except for two cases in RoNIN and RIDI datasets. R-TCN can get a smaller T-RTE number than CTIN in the RIDI-unseen test case; R-ResNet reports the smallest ATE of 3.99 for RoNIN-seen. In particular, CTIN improves an average ATE on all seen test datasets by 34.74%, 21.78%, and 37.46% over R-LSTM, R-ResNet, and R-TCN, respectively; the corresponding numbers for all unseen test datasets are 34.73%, 3.97%, and 29.47%.

The main limitation of RoNIN variants (*i.e.,* R-LSTM, R-ResNet, and R-TCN) is that they do not capture the spectral correlations across time-series which hampers the performance of the model. Therefore, it is convincing that CTIN achieves better performance over these baselines. Table 2 also shows that CTIN generalizes well to unseen test sets, and outperforms all other models on test sets. PDR shows a persistent ATE due to the consistent and precise updates owing to the jerk computations. This mechanism leads to PDR failure on long trajectories. Over time, the trajectory tends to drift owing to the accumulated heading estimation and the drift would increase dramatically, which results in decentralized motion trajectory shapes. R-LSTM does not show satisfactory results over large-scale trajectories. The margin of the outperforms of CTIN compared to R-LSTM and R-TCN is notable. The results for SINS show a large drift that highlights the noisy sensor measurements from smartphones.

Ablation Study

Model Behaviors. The experimental results about performance comparisons between CTIN and three RoNIN variants are shown in Figure 2 and Table 3. In Figure 2a, each plot shows the cumulative density function (CDF) of the chosen metric on the entire test datasets. The blue line of CTIN is steeper than other plots, which indicates that CTIN shows significantly lower overall errors than all RoNIN variants for all presented metrics. As shown in Figure 2b, although CTIN's overall MSE is higher than R-Resnet and smaller than R-LSTM and R-TCN, its position drift error (*i.e.,* PDE (%)) is the smallest (*i.e.,* the best). In Table 3, we show the number of parameters for each model, GFLOPs performed by GPU during testing, the average GPU execution time for testing a sequence of IMU samples (excluding the time to load data and generate trajectories after model

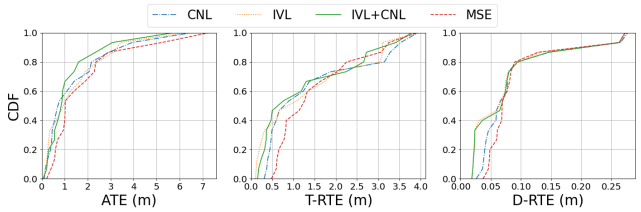


Figure 4: The performance of CTIN network with different loss functions evaluated on CTIN dataset.

Model	#Params (10^6)	GFLOP ($10^9/s$)	Avg. GPU time (ms)	Trajectory Error (meter)		
				ATE	T-RTE	D-RTE
CTIN	0.5571	7.27	65.96	**1.28**	**1.29**	**0.08**
R-LSTM	0.2058	7.17	704.23	2.22	2.10	0.11
R-TCN	2.0321	33.17	19.05	2.02	1.73	0.11
R-ResNet	4.6349	9.16	75.89	2.39	2.01	0.16

Table 3: Models' Evaluation Performance on CTIN dataset

prediction) and trajectory errors. Overall, CTIN possesses a significantly smaller number of parameters than R-TCN and R-ResNet, and more parameters than R-LSTM, achieving a competitive runtime performance with lower trajectory errors in a real deployment. Therefore, CTIN performs better than all RoNIN variants.

Attention Effectiveness. In this paper, we propose a novel attention mechanism to exploit local and global dependencies among the spatial feature space, and then leverage the multi-head attention layer to combine spatial and temporal information for better accuracy of velocity prediction. To evaluate their effectiveness, we conduct a group of experiments using CTIN/R-ResNet and their variant without/with the capability of attention mechanism. The experimental results are shown in Figure 3. Figure 3a shows that *CTIN-Atts* and *R-ResNet-Atts* models outperform the models without attention layer. Furthermore, *CTIN-Atts* perform the best for all metrics, and the performance of *CTIN-Conv* is better than all R-ResNet variants. In Figure 3b, *CTIN-Atts* and *R-ResNet-Atts* have lower average MSE loss of velocity prediction and smallest PDE than *CTIN-Conv* and *R-ResNet-Conv*. Overall, CTIN and R-ResNet can benefit from the proposed attention mechanism.

Loss function. In this section, we evaluate the performance of multi-task loss (*i.e.,* IVL+CNL) by performing a group comparison experiments using different loss functions, such as mean square error (MSE), Integral Velocity Loss (IVL) and Covariance NLL Loss (CNL), to train the models. As shown in Figure 4, CTIN with a loss of IVL+CNL achieves the best performance for ATE and D-RTE metrics.

Related Work

Conventional Newtonian-based solutions to inertial navigation can benefit from IMU sensors to approximate positions and orientations (Kok, Hol, and Schön 2017). In a strap-down inertial navigation system (SINS) (Savage 1998), accelerometer measurements are rotated from the body to the navigation frame using a rotation matrix provided by an integration process of gyroscope measurements, then subtracted the earth's gravity. After that, positions can be obtained from double-integrating the corrected accelerometer readings (Shen, Gowda, and Roy Choudhury 2018). However, the multiple integrations can lead to exponential error propagation. To compensate for this cumulative error, step-based pedestrian dead reckoning (PDR) approaches rely on the prior knowledge of human walking motion to predict trajectories by detecting steps, estimating step length and heading, and updating locations per step (Tian et al. 2015).

Data-Driven approach. Recently, a growing number of research works leverage deep learning techniques to extract information from IMU measurements and achieve competitive results in position estimation (Chen et al. 2018,?; Herath, Yan, and Furukawa 2020; Dugne-Hennequin, Uchiyama, and Lima 2021). IoNeT (Chen et al. 2018) first proposed an LSTM structure to regress relative displacement in 2D polar coordinates and concatenate to obtain the position. In RIDI (Yan, Shan, and Furukawa 2018) and RoNIN (Herath, Yan, and Furukawa 2020), IMU measurements are first rotated from the body frame to the navigation from using device orientation. While RIDI regressed a velocity vector from the history of IMU measurements to optimize bias, then performed double integration from the corrected IMU samples to estimate positions. RoNIN regressed 2D velocity from a sequence of IMU sensor measurements directly, and then integrate positions.

In addition to using networks solely for pose estimates, an end-to-end differentiable Kalman filter framework is proposed in Backprop KF (Haarnoja et al. 2016), in which the noise parameters are trained to produce the best state estimate, and do not necessarily best capture the measurement error model since loss function is on the accuracy of the filter outputs. TLIO provides a neural model to regress the velocity prediction and uncertainties jointly (Liu et al. 2020). In IDOL (Sun, Melamed, and Kitani 2021) two separate networks in an end-to-end manner are exploited. The first model is used to predict orientations to circumvent the inaccuracy in the orientation estimations with smartphone APIs. Next, the IMU measurements in the world frame are used to predict the velocities using the second model.

Conclusion

In this paper, we propose CTIN, a novel robust contextual Attention-based model to regress accurate 2D velocity from segments of IMU measurements. We first design a ResNet-based encoder to capture spatial contextual information from IMU measurements, further fuse these spatial representations with temporal knowledge by leveraging attention in the Transformer decoder. Finally, multi-task learning using uncertainty is leveraged to improve learning efficiency and prediction accuracy of 2D velocity. Through extensive experiments over a wide range of inertial datasets, CTIN is very robust and outperforms state-of-the-art models.

Acknowledgments

This material is based upon work supported by the U.S. Army Combat Capabilities Development Command Soldier Center (CCDC SC) Soldier Effectiveness Directorate (SED) SFC Paul Ray Smith Simulation & Training Technology Center (STTC) under contract No. W912CG-21-P-0009. Any opinions, findings, and conclusions, or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the CCDC-SC-SED-STTC.

References

Ahmetovic, D.; Gleason, C.; Ruan, C.; Kitani, K.; Takagi, H.; and Asakawa, C. 2016. NavCog: a navigational cognitive assistant for the blind. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*, 90–99.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Bloesch, M.; Omari, S.; Hutter, M.; and Siegwart, R. 2015. Robust visual inertial odometry using a direct EKF-based approach. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 298–304. IEEE.

Brajdic, A.; and Harle, R. 2013. Walk detection and step counting on unconstrained smartphones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, 225–234.

Chen, C.; Lu, X.; Markham, A.; and Trigoni, N. 2018. Ionet: Learning to cure the curse of drift in inertial odometry. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Dugne-Hennequin, Q. A.; Uchiyama, H.; and Lima, J. P. S. D. M. 2021. Understanding the Behavior of Data-Driven Inertial Odometry With Kinematics-Mimicking Deep Neural Network. *IEEE Access*, 9: 36589–36619.

Foxlin, E. 2005. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer graphics and applications*, 25(6): 38–46.

Haarnoja, T.; Ajay, A.; Levine, S.; and Abbeel, P. 2016. Backprop kf: Learning discriminative deterministic state estimators. In *Advances in neural information processing systems*, 4376–4384.

Han, K.; Wang, Y.; Chen, H.; Chen, X.; Guo, J.; Liu, Z.; Tang, Y.; Xiao, A.; Xu, C.; Xu, Y.; et al. 2020. A Survey on Visual Transformer. *arXiv preprint arXiv:2012.12556*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Herath, S.; Yan, H.; and Furukawa, Y. 2020. RoNIN: Robust Neural Inertial Navigation in the Wild: Benchmark, Evaluations, & New Methods. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 3146–3152. IEEE.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Hu, H.; Zhang, Z.; Xie, Z.; and Lin, S. 2019. Local relation networks for image recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3464–3473.

Kendall, A.; and Gal, Y. 2017. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30.

Kendall, A.; Gal, Y.; and Cipolla, R. 2018. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7482–7491.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kok, M.; Hol, J. D.; and Schön, T. B. 2017. Using inertial sensors for position and orientation estimation. *arXiv preprint arXiv:1704.06053*.

Leutenegger, S.; Lynen, S.; Bosse, M.; Siegwart, R.; and Furgale, P. 2015. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3): 314–334.

Li, J.; Guo, M.; and Li, S. 2017. An indoor localization system by fusing smartphone inertial sensors and bluetooth low energy beacons. In *2017 2nd International Conference on Frontiers of Sensors Technologies (ICFST)*, 317–321. IEEE.

Li, Y.; Yao, T.; Pan, Y.; and Mei, T. 2021. Contextual transformer networks for visual recognition. *arXiv preprint arXiv:2107.12292*.

Liu, W.; Caruso, D.; Ilg, E.; Dong, J.; Mourikis, A. I.; Daniilidis, K.; Kumar, V.; and Engel, J. 2020. TLIO: Tight Learned Inertial Odometry. *IEEE Robotics and Automation Letters*, 5(4): 5653–5660.

Lymberopoulos, D.; Liu, J.; Yang, X.; Choudhury, R. R.; Handziski, V.; and Sen, S. 2015. A realistic evaluation and comparison of indoor location technologies: Experiences and lessons learned. In *Proceedings of the 14th international conference on information processing in sensor networks*, 178–189.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32: 8026–8037.

Prechelt, L. 1998. Early stopping-but when? In *Neural Networks: Tricks of the trade*, 55–69. Springer.

Ramachandran, P.; Parmar, N.; Vaswani, A.; Bello, I.; Levskaya, A.; and Shlens, J. 2019. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*.

Russell, R. L.; and Reale, C. 2021. Multivariate uncertainty in deep learning. *IEEE Transactions on Neural Networks and Learning Systems*.

Santoro, A.; Raposo, D.; Barrett, D. G.; Malinowski, M.; Pascanu, R.; Battaglia, P.; and Lillicrap, T. 2017. A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*.

Savage, P. G. 1998. Strapdown inertial navigation integration algorithm design part 2: Velocity and position algorithms. *Journal of Guidance, Control, and dynamics*, 21(2): 208–221.

Shen, S.; Gowda, M.; and Roy Choudhury, R. 2018. Closing the gaps in inertial motion tracking. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*, 429–444.

Sturm, J.; Magnenat, S.; Engelhard, N.; Pomerleau, F.; Colas, F.; Cremers, D.; Siegwart, R.; and Burgard, W. 2011. Towards a benchmark for RGB-D SLAM evaluation. In *Rgb-d workshop on advanced reasoning with depth cameras at robotics: Science and systems conf.(rss)*.

Sun, S.; Melamed, D.; and Kitani, K. 2021. IDOL: Inertial Deep Orientation-Estimation and Localization. *arXiv preprint arXiv:2102.04024*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112.

Tian, Q.; Salcic, Z.; Kevin, I.; Wang, K.; and Pan, Y. 2015. An enhanced pedestrian dead reckoning approach for pedestrian tracking using smartphones. In *2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, 1–6. IEEE.

Titterton, D.; Weston, J. L.; and Weston, J. 2004. *Strapdown inertial navigation technology*, volume 17. IET.

Usenko, V.; Engel, J.; Stückler, J.; and Cremers, D. 2016. Direct visual-inertial odometry with stereo cameras. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 1885–1892. IEEE.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, D.; Li, Y.; Wang, L.; and Gong, B. 2020. Neural networks are more productive teachers than human raters: Active mixup for data-efficient knowledge distillation from a blackbox model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1498–1507.

Wang, X.; Girshick, R.; Gupta, A.; and He, K. 2018. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 7794–7803.

Wu, Y.; Schuster, M.; Chen, Z.; Le, Q. V.; Norouzi, M.; Macherey, W.; Krikun, M.; Cao, Y.; Gao, Q.; Macherey, K.; et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Yan, H.; Shan, Q.; and Furukawa, Y. 2018. RIDI: Robust IMU double integration. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 621–636.

Yang, Y.; Xing, W.; Wang, D.; Zhang, S.; Yu, Q.; and Wang, L. 2021. AEVRNet: Adaptive exploration network with variance reduced optimization for visual tracking. *Neurocomputing*, 449: 48–60.

Yao, J.; Wang, D.; Hu, H.; Xing, W.; and Wang, L. 2022. ADCNN: Towards learning adaptive dilation for convolutional neural networks. *Pattern Recognition*, 123: 108369.

Yao, J.; Xing, W.; Wang, D.; Xing, J.; and Wang, L. 2021. Active dropblock: Method to enhance deep model accuracy and robustness. *Neurocomputing*, 454: 189–200.

Zhang, J.; and Singh, S. 2014. LOAM: Lidar Odometry and Mapping in Real-time. In *Robotics: Science and Systems*, volume 2.

Zhao, H.; Jia, J.; and Koltun, V. 2020. Exploring self-attention for image recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10076–10085.