# Deep Neural Network Based Odometry Using Off-Shelf IMU

Naman Kaushik*, Prof. Rajesh Kumar*
*Mechanical Engineering, Indian Institute of Technology BHU Varanasi, India

***Abstract*—Modern MEMS Inertial Measurements Units (IMUs) are small, cheap, energy efficient, and widely employed in smart devices,indoor localization and mobile robots. This in turn lays foundation for pervasive Internet-of-Things applications and services.However, low-cost inertial sensors, as commonly found in smartphones, have bias and noise, which leads to unbounded growth in error when accelerations are double integrated to obtain displacement. Small errors in state estimation propagate to make odometry virtually unusable in a matter of seconds.Recently, there has been a growing interest in applying deep neural networks (DNNs) to motion sensing and location estimation. In this project we propose a transformer based deep learning framework for performing odometry by using only the inertial data obtained from a low-cost IMU installed in smartphones. Results obtained are are favorably comparable to the state-of-the-art inertial odometry techniques.**

***Index Terms*—Odometry, IMU, Deep Transformer Model.**

## I. Introduction

Odometry is a process to compute relative sensor pose changes between two sequential moments. This is generally essential for various applications that need to track the target device poses in a 3D unknown environment. Fast and accurate indoor localization is a fundamental need for many personal applications, including smart retail, public places navigation, human-robot interaction, and augmented reality. One of the most promising approaches is to use inertial sensors to perform dead reckoning, which has attracted great attention from both academia and industry, because of its superior mobility and flexibility.

Recent advances of MEMS (Micro-electro-mechanical systems) sensors have enabled inertial measurement units (IMUs) small and cheap enough to be deployed on smartphones. However, the low-cost inertial sensors on smartphones are plagued by high sensor noise, leading to unbounded system drifts. Based on Newtonian mechanics, traditional strap-down inertial navigation systems (SINS) integrate IMU measurements directly. They are hard to realize on accuracy-limited IMU due to exponential error propagation through integration. To address these problems, step-based pedestrian dead reckoning (PDR) has been proposed. This approach estimates trajectories by detecting steps, estimating step length and heading, and updating locations per step. Instead of double integrating accelerations into locations, a step length update mitigates exponential increasing drifts into linear increasing drifts. However, dynamic step estimation is heavily influenced by sensor noise, user's walking habits, and phone attachment changes, causing unavoidable errors to the entire system. In some scenarios, no steps can be detected, for example, if a phone is placed on a baby stroller or shopping trolley, the assumption of periodicity, exploited by step-based PDR would break down. Therefore, the intrinsic problems of SINS and PDR prevent widespread use of inertial localization in daily life. Many studies have been done to cure the unavoidable 'curse' of inertial systems drift, and break the cycle of continuous error propagation.

To solve this problem, machine learning based approaches have recently been introduced. For instance, the velocity of the IMU attached on a human body is regressed by using the velocity from visual odometry as a ground truth for the training process. Also, the location and direction on a 2D floor map can be regressed with a deep learning framework. Estimating magnitude of translation and rotation changes using deep neural networks can also be performed.

In this project we propose a transformer based deep neural network (DNN) architecture which have advantages over the traditional long short-term memory (LSTM) models. To the best of our knowledge this is the first DNN based odometry solution using transformers.

## II. Related Work

In this section, we provide a brief overview of some related work in inertial navigation systems, pedestrian dead reckoning (PDR), visual-inertial odometry, and deep learning.

**Strapdown Inertial Navigation System:** Strapdown inertial navigation systems (SINS) have been studied for decades. Previous inertial systems heavily relied on expensive, heavy, high-precision inertial measurement units, hence their main application had to be constrained on moving vehicles, such as automobiles, ships, aircraft, submarines, and spacecraft. Recent advances in MEMS technology enable low-cost MEMS IMU to be deployed on robotics, UAV, and mobile devices. However, restricted by size and cost, the accuracy of a MEMS IMU is extremely limited and has to be integrated with other sensors, such as visual-inertial odometry. Another solution is to attach an IMU on the user's foot in order to take advantage of heel strikes for a zero-velocity update to compensate system error drifts. These inconveniences prevent the wide adoption of inertial solutions on consumer-grade devices.

**Pedestrian Dead Reckoning:** Unlike SINS' open-loop integration of inertial sensors, PDR uses inertial measurements to detect steps, estimate stride length, and heading via an empirical formula. System errors still quickly accumulate, because of incorrect step displacement segmentation and inaccurate stride estimation. In addition, a large number of parameters have to be carefully tuned according to a user's walking habits. Recent research mainly focused on fusing

PDR with external references, such as a floor plan, WiFi fingerprinting, or ambient magnetic fields, still leaving the fundamental problem of rapid drift unsolved. Step-based PDR cannot address handle more general tracking problems, including trolley/wheeled configurations.

**Visual-inertial odometry:** One way to avoid the error accumulation in the IMU-based 6-DOF odometry is to use it in conjunction with a monocular camera. There are several 6-DOF visual-inertial odometry (VIO) methods available, such as VINS-MONO and OKVIS. Owing to the recent advance of deep learning, end-to-end approaches were also proposed, which employ a neural network to fuse visual and inertial data. PIVO is considered an inertial-visual odometry technique because a higher emphasis is given to the use of an IMU in order to be more robust to a lack of discriminative features in the images. This is accomplished by using an Extended Kalman Filter (EKF) estimation. Nevertheless, making use of visual odometry causes an increase of energy consumption and processing demands and a decrease in the accuracy under ill-conditioned surroundings.

**Deep Learning:** Recently, machine learning techniques have been applied to the purely inertial odometry problem, being able to obtain superior results, compared with SINS and PDR. RIDI estimated phone motion attached to a human body by first using a support vector machine to classify phone attachment locations such as leg, bag, hand or body, and then employing support-vector regression trained for the location of the given sensors to predict the device velocity. Such regressed velocity is finally used to correct accelerations on a 2D map, which are then double integrated to compute the odometry. This method is basically classified into PDR. IONet employed LSTM to obtain displacements in polar coordinates from the IMU data. However, this method also focused only on estimating the 2D planar trajectory as well as PDR, which is 3-DOF odometry. To the best of our knowledge there is no transformer based inertial odometry using a low-cost IMU in the literature.

## III. MODEL ARCHITECTURE

### A. *Time2Vector*

In order to overcome a Transformer's temporal indifferences, we will implement the approach described in the paper Time2Vec: Learning a Vector Representation of Time. The authors of the paper propose "a model-agnostic vector representation for time, called Time2Vec". You can think of a vector representation just like a normal embedding layer that can be added to a neural network architecture to improve a model's performance.
The paper provide the main idea that a meaningful representation of time has to include both periodic and non-periodic patterns. The time vector/representation t2v is comprised of two components, first component represents the non-periodic/linear and the second represents the periodic feature of the time vector.The authors experimented with different functions to

$$\mathbf{t2v}(\tau)[i] = \begin{cases} \omega_i \tau + \varphi_i, & \text{if } i = 0. \\ \mathcal{F}(\omega_i \tau + \varphi_i), & \text{if } 1 \le i \le k. \end{cases}$$

Fig. 1. Mathematical representation of time vector — Time2Vec: Learning a Vector Representation of Time

best describe a periodic relationship (sigmoid, tanh, ReLU, mod, triangle, etc.). In the end, a sine-function achieved the best and most stable performance (cosine achieved similar results).

### B. *Transformer*

A Transformer is a neural network architecture that uses a self-attention mechanism, allowing the model to focus on the relevant parts of the time-series to improve prediction qualities. The self-attention mechanism consists of a Single-Head Attention and Multi-Head Attention layer. The self-attention mechanism is able to connect all time-series steps with each other at once, lea ding to the creation of long-term dependency understandings. Finally, all these processes are parallelized within the Transformer architecture, allowing an acceleration of the learning process.

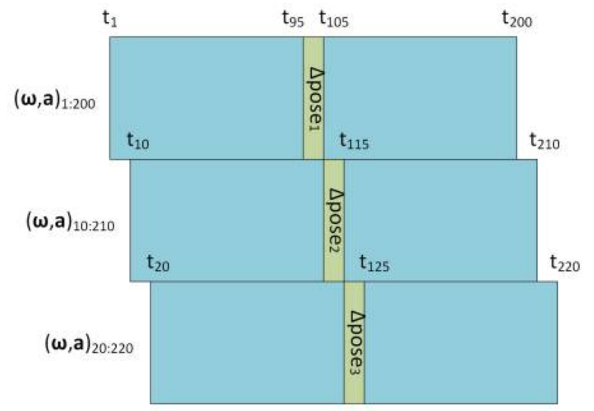


Fig. 2. Input and output on time-axis. IMU data windows are overlapped over time (blue), and both past and future frames are used when computing the relative pose at each pose moment (green).

*1) Single-Head Attention:* The single-head attention layer takes 3 inputs (Query, Key, Value) in total. For us, each Query, Key, and Value input is representative of the accelerometer and gyroscope data (with batch size of 32 and sequence length of 200 samples).

After the initial linear transformation, we will calculate the attention score/weights. The attention weights determine how much focus is placed on individual time-series steps when predicting a future stock price. Attention weights are calculated by taking the dot-product of the linearly transformed Query and Key inputs, whereas the transformed Key input has been transposed to make the dot-product multiplication feasible. Then the dot-product is divided by the dimension size of the previous dense layers, to avoid exploding gradients. The divided dot-product then goes through the softmax function to yield a set of weights that sum up 1. As the last step, the calculated softmax matrix which determines the focus of each time step is multiplied with the transformed v matrix which concludes the single-head attention mechanism.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$$

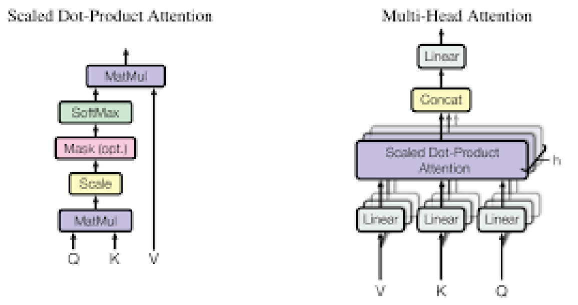


Fig. 3. Single and Multiple Head Attention Transformer Models.

*2) Multi-Head Attention:* To further improve the self-attention mechanism the authors of the paper **Attention Is All You Need** proposed the implementation of multi-head attention. The functionality of a multi-head attention layer is to concatenate the attention weights of n single-head attention layers and then apply a non-linear transformation with a Dense layer. The illustration below shows the concatenation of 3 single-head layers.
Having the output of n single-head layers allows the encoding of multiple independent single-head layers transformation into the model. Hence, the model is able to focus on multiple time-series steps at once. Increasing the number of attention heads impacts a model's ability to capture long-distance dependencies positively. (We have chosen *n=12*)

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_n)W^O$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$

*3) Transformer Encoder Layer:* The single- and multi-head attention mechanisms (self-attention) are now aggregated into a transformer encoder layer. Each encoder layer incorporates a self-attention sublayer and a feedforward sublayer. The feedforward sublayer consists of two dense layers with ReLU activation in between.Each sublayer is followed by a dropout layer, after the dropout, a residual connection is formed by adding the initial Query input to both sublayer outputs. Concluding each sublayer a normalization layer placed after the residual connection addition to stabilize and accelerate the training process. Three Transformer layers are stacked over the Time2Vec embedding in our model, it is to be noted that we are not using Transformer decoder layers our implemented Transformer architecture.

### C. Relative Pose Representation

There are several approaches to represent a 6-DOF relative pose. One approach is to simply extend the polar coordinate system to the 3D space by using the spherical coordinate system.This allows obtaining a correct trajectory.

$$x_t = x_{t-1} + \Delta l \cdot \sin(\theta_{t-1} + \Delta\theta) \cdot \cos(\phi_{t-1} + \phi)$$
$$y_t = y_{t-1} + \Delta l \cdot \sin(\theta_{t-1} + \Delta\theta) \cdot \cos(\phi_{t-1} + \phi)$$
$$z_t = z_{t-1} + \Delta l \cdot \cos(\theta_{t-1} + \Delta\theta)$$

However, one drawback is that the orientation will only be consistent when forward motion occurs. For example, if the system moves backwards or sideways with no change in the orientation, this will be interpreted as a forward movement together with a change of orientation in the backward or sideway direction.// Another approach is to use a 3D translation vector $\Delta p$ and a unit quaternion $\Delta q$.This representation correctly handles the orientation when dealing with motions in any direction.From a previous position $p_{t-1}$ and orientation $q_{t-1}$ , the current position $p_t$ and orientation $q_t$ after applying a pose change $(\Delta p, \Delta q)$ is given by

$$p_t = p_{t-1} + R(q_{t-1})\Delta p$$
$$q_t = q_{t-1} \otimes R$$

where **R(q)** is the rotation matrix for q, and $\otimes$ is the Hamilton product.

### D. Pose Distance Metric

A straightforward approach to estimate the difference between ground truth and predicted poses is to compute their Mean Square Error(MSE). This is done when using the spherical coordinates representation, namely MSE loss function. However, MSE is an algebraic rather than a geometric distance function.For the 6-DOF pose representation that employs quaternions, loss functions more related to the actual geometric difference between the ground truth pose ( p, q ) and the predicted pose ( p̂,q̂ ) can be defined. Therefore, one alternative is to use the absolute value of the 3D pose graph SLAM error metric. In this case, the corresponding loss function is Translative Mean Absolute Error(TMAE) and Quaternion Multiplicative Error(QME).

$$L_{TMAE} = ||\hat{p} - p||_1$$
$$L_{QME} = 2 \cdot ||imag(\hat{q} \otimes \bar{q})||_1$$

where imag(**q**) returns the imaginary part of **q** and $\bar{\mathbf{q}}$ is the complex conjugate of **q**.

### E. Multi-Task Learning for Metric Balancing

Inspired by *Multi-task learning using uncertainty*we aim to maximize the log Gaussian likelihood of the model, which is equivalent to minimizing the following final loss function:

$$L_{MTL} = \sum_{i=1}^{n} exp(-\log {\sigma_i}^2)L_i + \log {\sigma_i}^2$$

where ${\sigma_i}^2$and $L_i$ are the variance and the loss function for the $i^th$ task, respectively. The log variance values act as weights to the individual losses of each task. The goal of the training procedure is then to learn the $\log {\sigma_i}^2$ that minimizes $L_{MTL}$ from the ground truth data.The trainable weights of the multi-loss layer are the log variance values $\log {\sigma_i}^2$, each one associated to a layer input. All these weights are initially set to 0.The individual loss functions: $L_1 = L_{TMAE}$ and $L_2 = L_{QME}$.

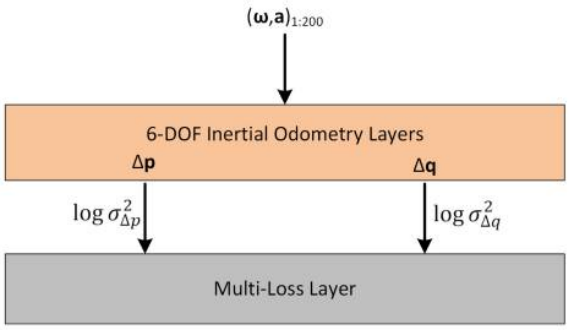
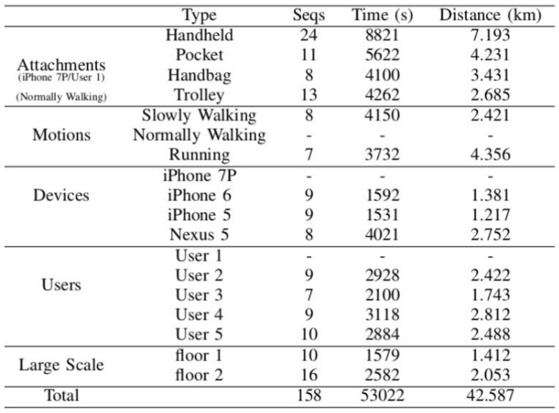

$(\omega,\mathbf{a})_{1:200}$
6-DOF Inertial Odometry Layers
$\Delta\mathbf{p}$
$\Delta\mathbf{q}$
$\log \sigma^2_{\Delta p}$
$\log \sigma^2_{\Delta q}$
Multi-Loss Layer

Fig. 4. Network architecture with multi-task learning for the translation vector with quaternion. The multi-loss layer allows learning the weights (log variances) associated to each of two tasks (translation and orientation change estimation).

## IV. Dataset

Experiments were performed using sequences obtained with a handheld smartphone, sequences from the Oxford Inertial Odometry Dataset (OxIOD) were used.



| | Type | Seqs | Time (s) | Distance (km) |
|---|---|---|---|---|
| Attachments (iPhone 7P/User 1) (Normally Walking) | Handheld | 24 | 8821 | 7.193 |
| | Pocket | 11 | 5622 | 4.231 |
| | Handbag | 8 | 4100 | 3.431 |
| | Trolley | 13 | 4262 | 2.685 |
| Motions | Slowly Walking | 8 | 4150 | 2.421 |
| | Normally Walking | - | - | - |
| | Running | 7 | 3732 | 4.356 |
| Devices | iPhone 7P | - | - | - |
| | iPhone 6 | 9 | 1592 | 1.381 |
| | iPhone 5 | 9 | 1531 | 1.217 |
| | Nexus 5 | 8 | 4021 | 2.752 |
| Users | User 1 | - | - | - |
| | User 2 | 9 | 2928 | 2.422 |
| | User 3 | 7 | 2100 | 1.743 |
| | User 4 | 9 | 3118 | 2.812 |
| | User 5 | 10 | 2884 | 2.488 |
| Large Scale | floor 1 | 10 | 1579 | 1.412 |
| | floor 2 | 16 | 2582 | 2.053 |
| Total | | 158 | 53022 | 42.587 |

Fig. 5. Oxford Inertial Odometry Dataset.

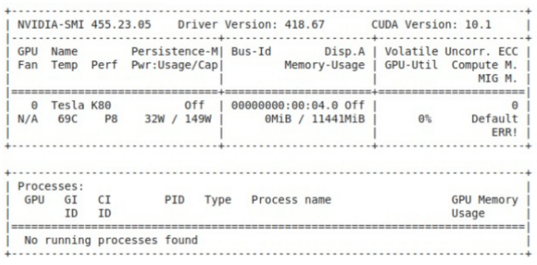
OxIOD provides angular velocity and linear acceleration data recorded with phones at a sampling rate of 100 Hz while moving around the environment under different conditions. It also contains precise and synchronized ground truth 6-DOF poses. The data collected from user1 holding an iPhone 7 Plus by hand while normally walking in a room was used, with a total of 24 sequences, a recording time of approximately 2 h and 27 min and a walking distance of 7.193 km. Due to the presence of noise in the ground truth measurements, the initial 12 s and the final 3 s of each sequence were discarded. From this data, 17 sequences were randomly chosen for the training and the remaining sequences were used for the testing. Training IMU data windows also have a stride of 10 frames, resulting in a total amount of 55,003 training samples. Excerpts of 20 s from the test sequences were employed for both qualitative and quantitative evaluations.

TABLE I
TEST TRAIN SPLIT OF OXIOD HANDHELD SYNC SEQUENCES

| Training Data | Test Data |
|---|---|
| data1/seq1 | data1/seq2 |
| data1/seq3 | data1/seq5 |
| data1/seq4 | data1/seq6 |
| data1/seq7 | data3/seq1 |
| data2/seq1 | data4/seq1 |
| data2/seq2 | data4/seq3 |
| data2/seq3 | data5/seq1 |
| data3/seq2 | |
| data3/seq3 | |
| data3/seq4 | |
| data3/seq5 | |
| data4/seq2 | |
| data4/seq4 | |
| data4/seq5 | |
| data5/seq2 | |
| data5/seq3 | |
| data5/seq4 | |

## V. Details of Training

Google Colab default GPU (Nvidia - Tesla) was used with tenforflow and keras as the backend. The Adam optimizer



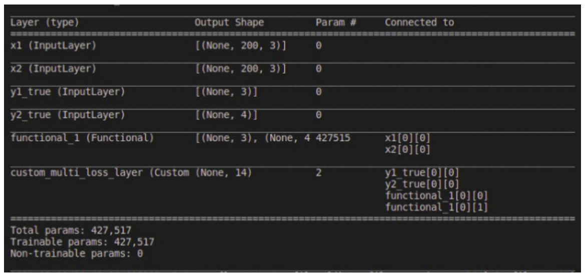
```
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 455.23.05    Driver Version: 418.67       CUDA Version: 10.1     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  Tesla K80           Off  | 00000000:00:04.0 Off |                    0 |
| N/A   69C    P8    32W / 149W |      0MiB / 11441MiB |      0%      Default |
|                               |                      |                 ERR! |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|  No running processes found                                                 |
+-----------------------------------------------------------------------------+
```

Fig. 6. Google Colab GPU Specifications.

was used with a learning rate of 0.0001, with batch size of 32 and sequence length of 200. Average time per epoch was 15 minutes. The model with best vaildation loss is kept after the training. Quantitative and qualitative evaluations were conducted in order to assess the effectiveness of the proposed inertial odometry solution.

We have used RMSE as the evaluation metric of our model.



```
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
x1 (InputLayer)                 [(None, 200, 3)]     0
__________________________________________________________________________________________________
x2 (InputLayer)                 [(None, 200, 3)]     0
__________________________________________________________________________________________________
y1_true (InputLayer)            [(None, 3)]          0
__________________________________________________________________________________________________
y2_true (InputLayer)            [(None, 4)]          0
__________________________________________________________________________________________________
functional_1 (Functional)       [(None, 3), (None, 4 427515      x1[0][0]
                                                                 x2[0][0]
__________________________________________________________________________________________________
custom_multi_loss_layer (Custom (None, 14)           2           y1_true[0][0]
                                                                 y2_true[0][0]
                                                                 functional_1[0][0]
                                                                 functional_1[0][1]
==================================================================================================
Total params: 427,517
Trainable params: 427,517
Non-trainable params: 0
```

Fig. 7.   Model Summary.

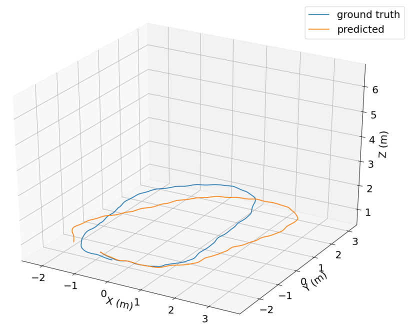


Fig. 8.   data1/seq2

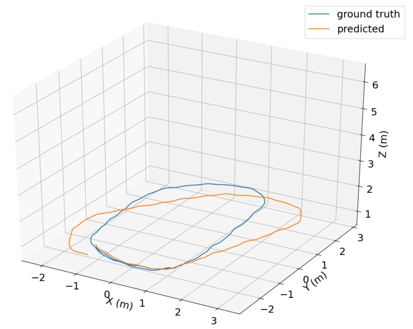


Fig. 9.   data1/seq5

## VI. Conclusion

The presented odometry solution is able to successfully provide 6-DOF relative pose estimates using solely noisy and biased data from the low-cost IMU installed in the smartphone. To the best of our knowledge this is the solution to use transformer architecture, transformers performs parallel computation in contrast to the LSTMs which perform sequential computation and hence produce results relatively faster. Moreover transformers utilizes positional encodings which which provides long-term dependency understandings which not available even in LSTM version of RNN architecture.

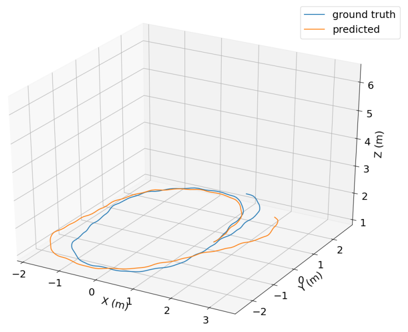


Fig. 10.   data1/seq6

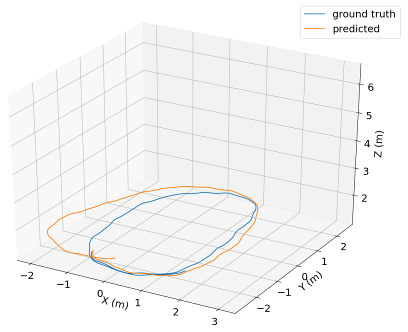


Fig. 11.   data3/seq1

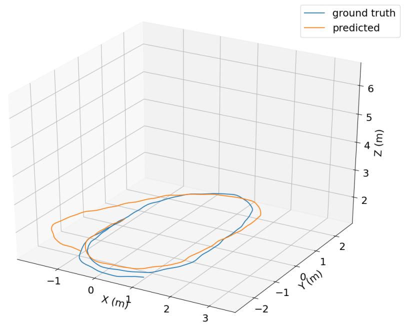


Fig. 12.   data4/seq1

Although this project represents only the example pedestrian tracking, it has been confirmed that the same model can be used on other patterns and datasets like the EuRoC MAV dataset. // Noticeable challenges during the project were requirement of GPU and high training time. Published results from the actual all the proposed odometry solutions were

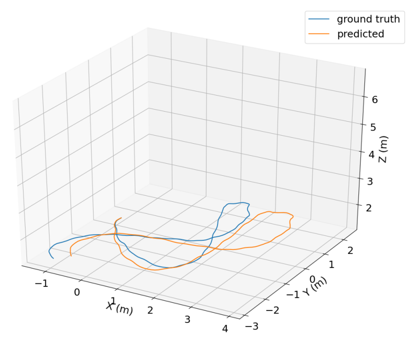


Fig. 13. data4/seq3

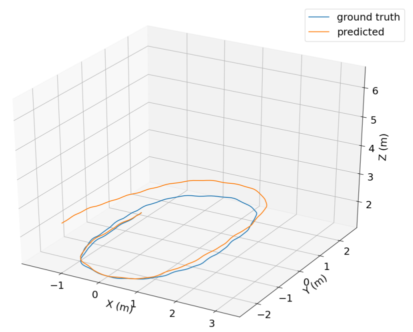


Fig. 14. data5/seq1

TABLE II
RMSE ON THE TEST SEQUENCES

| Test Sequence | RMSE |
|---|---|
| data1/seq2 | 1.543305 |
| data1/seq5 | 1.621635 |
| data1/seq6 | 0.719018 |
| data3/seq1 | 0.808223 |
| data4/seq1 | 0.785921 |
| data4/seq3 | 1.019505 |
| data5/seq1 | 0.620621 |

not available to perform relative performance study. It is also worth noting that our model never reached the saturation and hence requires more training time which was not possible with the current available hardware. // Although OxIOD also provides magnetometer data, from our experience such information is less reliable due to noise caused by magnetic fields from electrical devices.For different pattern we have train the model again on the dataset although transfer learning from can be used to use the learnings from similar patterns.

## REFERENCES

1 C. Chen, X. Lu, A. Markham, and N. Trigoni. *Ionet: Learning to cure the curse of drift in inertial odometry*. in Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, New Orleans, Louisiana, USA, February 2-7, 2018, 2018

2 Esfahani, M.A.; Wang, H.; Wu, K.; Yuan, S. *AbolDeepIO: A Novel Deep Inertial Odometry Network for Autonomous Vehicles.* IEEE Trans. Intell. Transp. Syst. 2019.

3 João Paulo Silva do Monte Lima , Hideaki Uchiyamaand Rin-ichiro Taniguchi. *End-to-End Learning Framework for IMU-Based 6-DOF Odometry*

4 Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, *Atten- tion is all you need.* In Guyon, I., Luxburg, U. V., Ben- gio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), Advances in Neural Information Pro- cessing Systems 30, pp. 5998–6008. Curran Associates, Inc., 2017.

5 Neo Wu, Bradley Green,Xue Ben and Shawn O'Banion. *Deep Transformer Models for Time Series Forecasting: The Influenza Prevalence Case.* arXiv Preprint 23 Jan 2020, arXiv:2001.08317v1.

6 Chen, C.; Zhao, P.; Lu, C.X.; Wang, W.; Markham, A.; Trigoni, N. *OxIOD: The Dataset for Deep Inertial Odometry.* arXiv Preprint 2018, arXiv:1809.07491.

7 Burri, M.; Nikolic, J.; Gohl, P.; Schneider, T.; Rehder, J.; Omari, S.; Achtelik, M.W.; Siegwart, R. *The EuRoC micro aerial vehicle datasets.* Int. J. Robot. Res. 2016, 35, 1157–1163.

8 Titterton, D.; Weston, J.L.; Weston, J. *Strapdown Inertial Navigation Technology;* IET: London, UK, 2004; Volume 17.

9 Yan, H.; Shan, Q.; Furukawa, Y. *RIDI: Robust IMU double integration. In Proceedings of the European Conference on Computer Vision,* Munich, Germany, 8–14 September 2018; pp. 621–636.

10 Kendall, A.; Gal, Y.; Cipolla, R. *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, Salt Lake City, UT, USA, 18–22 June 2018; pp. 7482–7491.

11 Seyed Mehran Kazemi , Rishab Goel , Sepehr Eghbali , Janahan Ramanan, Jaspreet Sahota, Sanjay Thakur, Stella Wu, Cathal Smyth, Pascal Poupart and Marcus Brubaker. *Time2Vec: Learning a Vector Representation of Time.* arXiv Preprint 11 Jul 2019, arXiv:1907.05321v1.

12 Ceres Solver. Available online: http://ceres-solver.org