# Improved State Propagation through AI-based Pre-processing and Down-sampling of High-Speed Inertial Data

Jan Steinbrener[1], Christian Brommer, Thomas Jantos, Alessandro Fornasier and Stephan Weiss

*Abstract*— **We present a novel approach to improve 6 degree-of-freedom state propagation for unmanned aerial vehicles in a classical filter through pre-processing of high-speed inertial data with AI algorithms. We evaluate both an LSTM-based approach as well as a Transformer encoder architecture. Both algorithms take as input short sequences of fixed length $N$ of high-rate inertial data provided by an inertial measurement unit (IMU) and are trained to predict in turn one pre-processed IMU sample that minimizes the state propagation error of a classical filter across $M$ sequences. This setup allows us to provide sufficient temporal history to the networks for good performance while maintaining a high propagation rate of pre-processed IMU samples important for later deployment on real-world systems. In addition, our network architectures are formulated to directly accept input data at variable rates thus minimizing necessary data preprocessing. The results indicate that the LSTM based architecture outperforms the Transformer encoder architecture and significantly improves the propagation error even for long IMU propagation times.**

## I. INTRODUCTION

Real-time, accurate and robust state estimation is a prerequisite for higher level autonomy in mobile robotic systems and has been the focus of research efforts for many years. For unmanned aerial vehicles (UAVs), Kalman filter based visual-inertial odometry (VIO) frameworks have been shown to provide accurate state estimation at required rates despite the low computational resources available [1]. These filter based approaches rely on inertial measurements provided by low-cost inertial measurement units (IMUs) to propagate the state in between lower-rate sensor measurements (e.g. from an onboard camera or global navigation satellite system (GNSS)). Due to the inherent noise in the IMU readings, IMU propagation suffers from significant drift and the resulting position estimate quickly degrades with time. This is true even if the noise parameters of the IMU have been accurately determined, a consequence of the random walk like behavior of the underlying stochastic processes.

While significant advances have been achieved in improving the robustness of the filters with respect to low quality sensor readings [2][3], enabling mid-air initialization of filters as well as run-time modularity [4], and automatically assessing filter performance and consistency, extensive efforts on mitigating the error during IMU propagation have only recently come into focus. With the advent of deep learning methods, powerful tools exist nowadays to learn non-linear behavior from noisy, heterogeneous data provided that enough (ground-truth labeled) data is available for training and that the training data reflect the expected variability of real-world scenarios [5]. Recurrent neural networks, in particular long short-term memory networks (LSTMs) [6], have been proven effective in extracting information from temporal sequences without suffering from vanishing gradients. Recently, Transformers, purely convolutional neural network (CNN) based architectures employing the concept of (self-) attention [7], have come into focus for processing temporal sequences - from applications in natural language processing [8] to analyzing sequences of images [9].

We present here a novel approach to improve 6 degree-of-freedom (DoF) state propagation for unmanned aerial vehicles in a classical filter through pre-processing of high-speed inertial data with state-of-the-art AI algorithms. By "pre-processing" we understand the processing of raw IMU data with AI-based models which yield "cleaner" pre-processed IMU data that are then used subsequently for propagation in a classical filter. Our contributions are as follows:

- Formulation and training of neural networks for pre-processing short IMU sequences at variable rates
- Neural network based down-sampling of high-rate IMU data
- Benchmark of two different neural network models compared to standard IMU integration
- Improved 6 degree-of-freedom state propagation for unmanned aerial vehicles
- Formulation of a least squares approach acting on motion data to refine estimated transformations between different IMU frames and to estimate their biases.

## II. RELATED WORK

Classical, robust approaches for 6-DoF state estimation combine IMU measurements with additional sensors and are either based on recursive filters [10], [1], optimization techniques [11] or a combination thereof. To improve performance, most classical methods focus on robust initialization methods [12], improved sensor processing algorithms [3], modularity with respect to sensor updates [4]. or efficient pre-integration methods for IMU data [13]. In almost all approaches, IMU integration is performed using first or higher order integration of the underlying equations of motion (dead reckoning). Recently, AI-based approaches have been applied successfully to learn from IMU data. Several groups have

[1]All authors are with the Control of Networked Systems group, Universität Klagenfurt, Lakeside Park B04a, 9020 Klagenfurt, Austria {jan.steinbrener christian.brommer, thomas.jantos, alessandro.fornasier, stephan.weiss}@aau.at

This work was supported by the Federal Ministry for Climate Action, Environment, Energy, Mobility, Innovation and Technology (BMK) under the grant agreements 878646 (AIMRobot) and 881082 (MUKISANO).

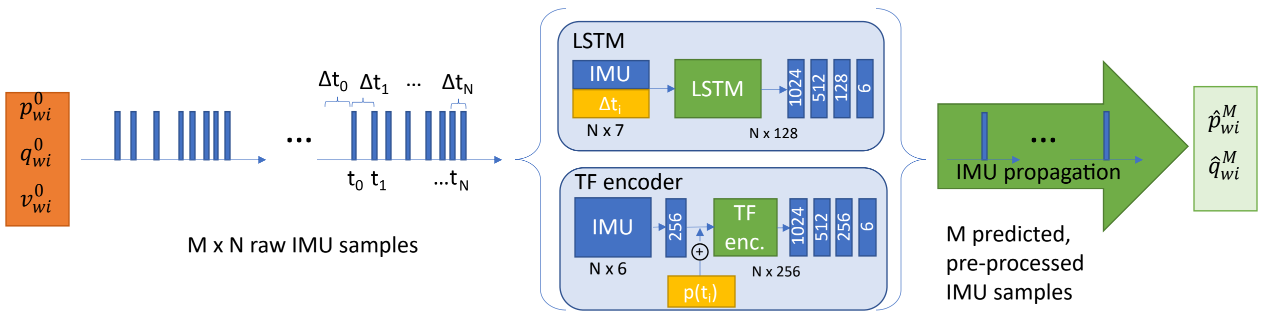

Fig. 1. Architecture and data handling of the proposed networks during training.

focused on IMU based attitude estimation. In [14], the authors performed denoising of IMU data for open-loop attitude estimation using a CNN-based architecture while Weber et al. [15] compared conventional filters with neural network-based filters for IMU based attitude estimation. In a separate study [16], the same authors proposed their own framework for attitude estimation based on recurrent neural networks with domain specific adaptations. With an extension to localization, Sun et al. [17] showed that recurrent neural networks can be applied in combination with classical filter frameworks to estimate the orientation and subsequently position of a smartphone to high accuracy. Hu et al. [18] trained a deep reinforcement learning network to estimate orientation from inertial data, outperforming traditional methods and providing an estimate on the error bound of the method. Liu et al. [19] show in offline experiments that an LSTM-based architecture trained with data collected from a real system can outperform classical methods for UAV attitude estimation.

Other studies focused on predicting noise covariance terms based on processing IMU data. Brossard et al. [20] used a neural network approach to adapt the covariance terms for the velocity state based on IMU readings in an automonous driving task. In [21], the authors have trained a CNN to implement an adaptive Kalman filter for a GNSS-based inertial navigation system for UAVs while in [22], a similar paradigm based on multi-task learning was explored for land based vehicles. This concept was extended to a magneto-inertial filter framework for indoor localization in [23] which used an LSTM to dynamically adapt the process noise covariance based on inertial data. Using a different learning method of Gaussian Variational Inference, Wong et al. [24] successfully learned the covariances for both sensors and motion models for vehicle trajectory estimation.

A study most closely related to our approach has been presented recently in [25]. Here the authors also relied on a recurrent neural network architecture to pre-process the IMU data for subsequent propagation in classical filter frameworks. In their study, the neural network also outperformed other methods but to our knowledge did not perform down-sampling of IMU data and was trained for much longer sequences (up to full trajectories), leaving open the performance on short IMU sequences required for real-time applications. In contrast, we have focused on models that have been trained exclusively with short sequences of IMU data to optimize the pre-processing for time scales relevant for real-world applications.

## III. NOTATION

In this work we use the following notation: ${}_A\boldsymbol{x}$ denotes a vector in frame of reference $\{A\}$, and ${}_A\boldsymbol{p}_{AB}$ denotes the displacement vector between $\{A\}$ and $\{B\}$, expressed in frame of reference $\{A\}$. This can also be expressed as $\boldsymbol{p}_{AB}$ for short. Further, $\mathbf{R}_{AB}$ denotes the rotation matrix employing the following rotation ${}_A\boldsymbol{x} = \mathbf{R}_{AB}\,{}_B\boldsymbol{x}$. Finally, ${}_A\boldsymbol{x}_m$ denotes a measured physical quantity expressed in frame of reference $\{A\}$.

## IV. METHODS

### A. Network models

Two different type of network models were trained and compared to standard IMU propagation: (i) an LSTM-based model and (ii) a Transformer encoder based model. The former consisted of a bi-layer, single directional LSTM model with hidden size of 128. The latter consisted of a transformer encoder stage with 3 encoder layers, 8 attention heads and 256 features as encoder inputs. The output of both models was then decoded by simple feed-forward networks consisting of four fully-connected layers (see Fig. 1 for details). Both models accept as input a sequence of $N$ raw IMU samples and return one pre-processed IMU sample ($[\omega, \mathbf{a}]$) after the decoder stage resulting in a down-sampling of the raw IMU data rate by a factor $N$. For the transformer encoder, the input IMU data was mapped into a higher dimensional space using a fully-connected layer followed by positional encoding (as suggested by e.g. [7]) before feeding it to the encoder. To account for the variable IMU rate, the networks also take information on the time between the raw IMU samples as input. For the LSTM, the vector of relative time differences between two subsequent raw IMU samples ($\boldsymbol{\Delta}\mathbf{t}$) is concatenated with the raw IMU measurements and passed directly to the LSTM. For the Transformer encoder, a vector of relative times ($t_i$ starting at $t_0 = 0$ for the first IMU sample of a given sequence) is used in the positional encoding such that

$$p_{i,d} = \begin{cases} sin(\frac{t_i * f}{10000^{d/D}}) & \text{if } d \text{ even} \\ cos(\frac{t_i * f}{10000^{d/D}}) & \text{if } d \text{ odd} \end{cases} \tag{1}$$

Here, $t_i$ is the relative timestamp of IMU sample $i$ in the sequence and $f$ is a scaling factor to yield meaningful differences in the resulting embedding frequencies (in our experiments $f = 10000$).

## B. IMU propagation

The resulting pre-processed IMU sequence is then integrated according to the state propagation performed in classical filter frameworks (see e.g. [26]) with the state dynamics given as

$$\dot{\mathbf{p}}_{wi} = \mathbf{v}_{wi} \tag{2}$$

$$\dot{\mathbf{v}}_{wi} = \mathbf{R}_{q_{wi}}(\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a) - \mathbf{g} \tag{3}$$

$$\dot{\mathbf{q}}_{\mathbf{wi}} = \frac{1}{2}\mathbf{\Omega}(\boldsymbol{\omega}_m - \mathbf{b}_w - \mathbf{n}_w)\mathbf{q}_{\mathbf{wi}} \tag{4}$$

where $\mathbf{p}_{wi}$ is the translation from the world to the IMU/body frame expressed in the world frame, $\mathbf{v}_{wi}$ is the corresponding velocity, $\mathbf{q}_{\mathbf{wi}}$ is the orientation of the IMU in the world frame, $\mathbf{a}_m$ is the measured acceleration in the IMU frame, $\mathbf{b}_a$ and $\mathbf{n}_a$ are the accelerometer bias and noise parameters, $\mathbf{g}$ is the gravity vector in the world frame, $\boldsymbol{\omega}_m$ is the measured angular velocity in the IMU frame, $\mathbf{b}_\omega$ and $\mathbf{n}_\omega$ are the gyro bias and noise parameters, and $\mathbf{\Omega}(\omega)$ is the quaternion multiplication matrix of $\omega$.

In particular, we use the state propagation subroutine of our modular sensor fusion framework [4] for the integration. Since we do not perform any updates, the IMU biases $\boldsymbol{b}_\omega$ and $\boldsymbol{b}_a$ are assumed to be constant and set to either the actual estimated values when propagating raw IMU samples or to zero when propagating pre-processed IMU data. The numerical integration of the IMU samples is carried out in first order approximation.

## C. Data recording and alignment

For training of the networks and evaluation of the propagation error, 19 different trajectories were recorded. The trajectories all included take-off and landing but varied in their characteristic maneuvers that were flown. Some trajectories included only a certain type of pattern (e.g. circles at a certain speed or horizontal or vertical squares) while some were executed more or less randomly at different speeds to excite all axes of the IMU. The shape of the trajectories was determined empirically. The completeness of the training data with respect to possible UAV modes can be analysed by means of an observability analysis. This is left for future work. Examples of different trajectories are shown in Fig. 2.

The data was recorded using a TWINs Science Copter platform with a Pixhawk PX4 autopilot which was flown manually in our drone hall equipped with a motion capture (MoCap) system collecting ground-truth position and attitude at 360 Hz. The high speed inertial measurement unit (IMU) (LSM9DS1 by ST Microelectronics) was attached to one of the arms of the copter (see Fig. 3). This IMU is capable

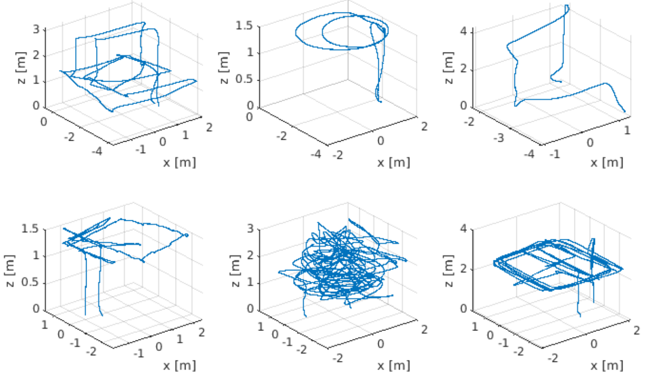

Fig. 2. Examples of recorded training trajectories

of recording inertial measurements at almost 1000 Hz at variable rates (nominal output data rate 952 Hz). In our experiments, the IMU rate varied between 800 and 900 Hz. To capture all of the flight dynamics, the range of the IMU was set to $\pm 16g$. The raw counts were converted to acceleration using the typical sensitivity provided by the manufacturer. For the angular velocities, the raw counts were converted to radians per second by multiplying with the corresponding sensitivity defined as the maximum scale of the gyro of ($\pm 245$ degree per second) divided by $2^{16} - 1$ and converting to radians. To determine the constant biases of the IMU, a couple of datasets were recorded with the platform in both, steady state and in motion with a sufficient motion excitement. The bias for the angular velocities $\boldsymbol{b}_\omega$ was determined as the mean of the measured angular velocities for the steady state dataset, whereas the bias for the linear accelerations $\boldsymbol{b}_a$ was determined by the solution of the following alignment problem. Let $\{P\}$ and $\{L\}$ denote frames of reference of two IMUs (PX4 and LSM). The transformation between the different IMUs, denoted $\mathbf{T}_{PL}$, were computed in two ways: firstly, we made use of Kalibr [27] with the help of an onboard camera. Secondly, we validate the computed transformation and compute the accelerometer bias by solving the following least-square problems.

$$\min_{\mathbf{R}_{PL}} \quad \|\mathbf{R}_{PL}\left({}_L\boldsymbol{\omega}_m - {}_L\boldsymbol{b}_\omega\right) - \left({}_P\boldsymbol{\omega}_m - {}_P\boldsymbol{b}_\omega\right)\|^2,$$

where $\mathbf{R}_{PL}$ denotes the rotation existing between the IMUs. Once the rotation has been computed we built a second least-square problem that allow us to determine the translation between the two IMUs as well as the accelerometer biases. In particular, differentiating the laws of motion yields

$$\mathbf{R}_{PL}\left({}_L\boldsymbol{a}_m - {}_L\boldsymbol{b}_a\right) = \left({}_P\boldsymbol{a}_m - {}_P\boldsymbol{b}_a\right) + \left(\lfloor {}_P\boldsymbol{\omega}_m - {}_P\boldsymbol{b}_\omega \rfloor_\times^2 + \lfloor {}_P\boldsymbol{\alpha} \rfloor_\times\right) {}_P\boldsymbol{p}_{PL},$$

where $\lfloor \cdot \rfloor_\times$ represent the skew-symmetric matrix, and ${}_P\boldsymbol{\alpha} = \frac{\mathrm{d}}{\mathrm{d}t}\,{}_P\boldsymbol{\omega}$ represent the angular acceleration, computed by numeric differentiation of the filtered (with a Savitzky–Golay filter) angular velocity ${}_P\boldsymbol{\omega} = {}_P\boldsymbol{\omega}_m - {}_P\boldsymbol{b}_\omega$.

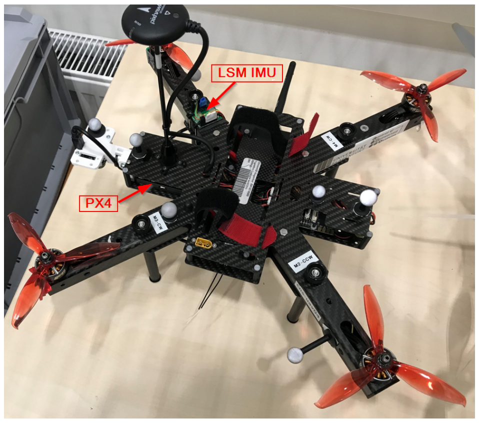

Fig. 3. Quadcopter with locations of LSM IMU and PX4.

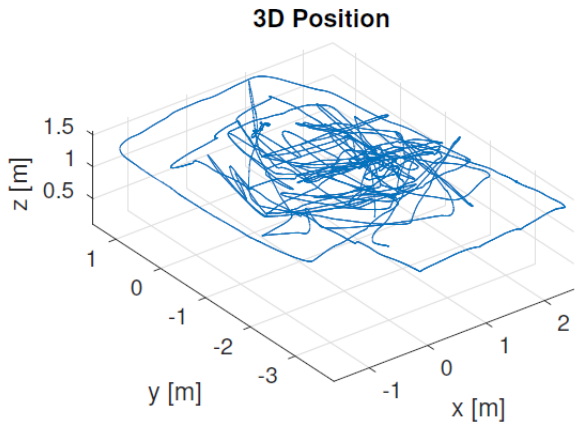

Fig. 4. Trajectory of the validation dataset.

Therefore we built the following least-square problem

$$\begin{aligned}\min_{\boldsymbol{x}} \quad & \|\boldsymbol{y} - \boldsymbol{\Xi}\,\boldsymbol{x}\|^2, \\ \text{s.t.} \quad & \boldsymbol{y} = \mathbf{R}_{PL}\,{}_L\boldsymbol{a}_m - {}_P\boldsymbol{a}_m, \\ & \boldsymbol{x} = \begin{bmatrix} {}_P\boldsymbol{p}_{PL} & {}_L\boldsymbol{b}_a & {}_P\boldsymbol{b}_a \end{bmatrix}^T, \\ & \boldsymbol{\Xi} = \begin{bmatrix} \lfloor {}_P\boldsymbol{\omega}_m - {}_P\boldsymbol{b}_\omega \rfloor_\times^2 + \lfloor {}_P\boldsymbol{\alpha} \rfloor_\times & -\mathbf{R}_{PL} & \mathbf{I} \end{bmatrix}.\end{aligned}$$

The resulting refined transformation is then used to align the ground-truth data with the LSM IMU for training and evaluation of the networks. This was necessary as, in our setup, the position of the copter in the world coordinate recorded by the MoCap system was aligned with the PX4 on the copter. In addition, the ground-truth pose data was interpolated to the same time stamps as the raw IMU data. For the position, a quadratic interpolation was performed while for the attitude expressed as quaternions, a spherical linear interpolation (SLERP [28]) was performed based on the linearized fractional time difference of the IMU sample at hand and the neighboring (in time) measured ground-truth attitudes.

### D. Network Training and Evaluation

For training the networks, 16 out of the available 19 datasets were used. The training datasets were chosen to represent a mixture of simple trajectories with excitation of one or few axes and more or less random trajectories with aggressive excitation of all axes to capture the range of possible scenarios. The number of IMU samples per IMU sequence was set to $N = 8$ resulting in an 8-fold downsampling of the raw IMU data rate. This number was chosen as a trade-off between providing temporal history to the networks and resulting rate of the pre-processed IMU data. Waiting for more raw IMU samples to be collected before pre-processing them with the AI model increases the downsampling rate but would also slow down the rate of pre-processed IMU data that is fed to the classical filter for propagation. The latter should still be high enough to be of practical consequence when implemented on a real system. With $N = 8$, the hypothetical rate of the processed IMU data is around 100 Hz which is still high enough for real-time 6-DoF pose estimation for dynamic systems such as UAVs.

A practical challenge in using short IMU sequences is that the accumulated pose error over the sequence length (here about 0.01 s) will be very small. This results in small loss values and makes training of the networks difficult. To overcome this problem, we chose to pass $M$ consecutive sequences of $N$ IMU samples through the networks before computing the position error and back-propagating the loss. Thus, the resulting graphs of the operations in the forward passes included $M$ calls to either the LSTM or Transformer encoder networks. Likewise, the gradients were propagated through $M$ instances of the networks during the backpropagation of the loss. In practice, we have found that this setup yielded reasonable loss values and did not suffer from vanishing gradients as one might assume. In our experiments we have set $M = 20$, i.e. the networks were trained with the position error evaluated after 20 predicted, pre-processed IMU samples (or $20 \times 8$ processed raw IMU samples). With this, we obtain in total $148,667$ IMU sequences of length $M \times N$ for training the networks.

We like to point out that passing multiple IMU sequences through the networks is only used for training. For testing or later deployment, the networks only take one sequence of length $N$ thus keeping the hypothetical inference rate of $1/N$-th of the raw IMU data rate.

The loss function was a combination of position loss and attitude loss

$$l = l_{\text{pos}} + l_{\text{rot}}, \quad \text{with} \tag{5}$$

$$l_{\text{pos}} = \phi_{L1}\left(\Delta\hat{\mathbf{p}}_{\mathbf{wi}}^{(\mathbf{M})}, \Delta\mathbf{p}_{\mathbf{wi}}^{(\mathbf{M})}\right) \tag{6}$$

$$l_{\text{rot}} = \phi_{L1}\left(\hat{\mathbf{R}}^{(\mathbf{M})}\mathbf{R}^{\mathbf{T},(\mathbf{M})}, \mathbf{I}_3\right) \tag{7}$$

where $\phi_{L1}$ is the smooth L1 loss function, $\Delta\hat{\mathbf{p}}_{\mathbf{wi}}^{(\mathbf{M})}$ is the estimated change in position after IMU propagation through $M$ sequences and $\Delta\mathbf{p}_{\mathbf{wi}}^{(\mathbf{M})}$ is the actual position change, $\hat{\mathbf{R}}^{(\mathbf{M})}$ and $\mathbf{R}^{\mathbf{T},(\mathbf{M})}$ are the estimated and actual relative rotation after $M$ sequences, respectively, and $I_3$ is

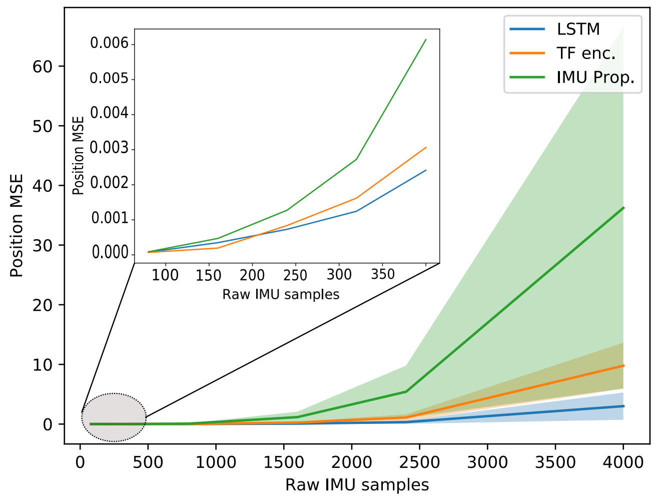

Fig. 5. Mean (solid line) and standard deviation (shaded area) of position MSE $[m^2]$ as a function of number of processed raw IMU samples. Length of sequence shown corresponds to about 5 seconds. Inset shows mean of MSE for the short propagation times of up to 0.5 seconds. The mean error for the LSTM at this point is about 4 cm.

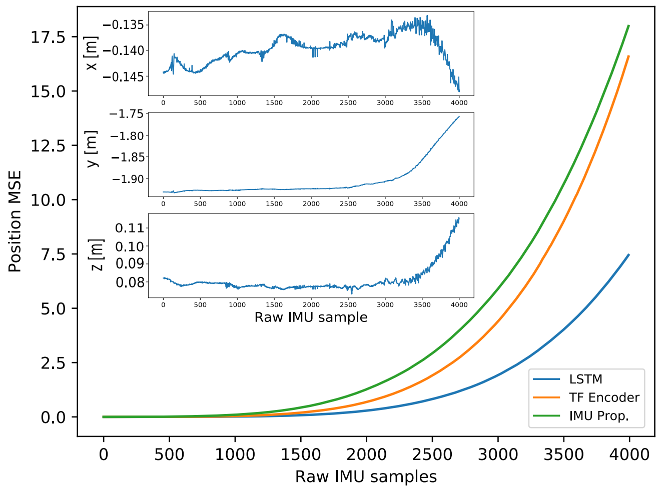

Fig. 6. Mean square error of position $[m^2]$ as a function of raw IMU samples processed starting from well defined initial conditions (copter at rest). Inset shows ground truth trajectory for same segment of IMU samples. Length of sequence shown corresponds to about 5 seconds.

the identity matrix in three dimensions. The loss function plays an important part in network training. While we have achieved good results with this loss function, future research will focus on more carefully balancing the positional and rotational loss terms and on improving the bias towards early rotational errors in the sequence. All networks were trained with a mini-batch size of 32 for 40 epochs using the Adam optimizer with a constant learning rate of $10^{-4}$. These hyperparameters were empirically determined to yield the best results. Contiguous samples of length $M \times N$ were drawn from random positions within the training datasets. Before propagating, ground truth information on the position, the attitude, and the initial velocity were injected into the state so that only the error after propagation of $M$ sequences was evaluated. The initial velocity was computed on the fly from the preceeding segment of $N$ IMU raw data samples as

$$\mathbf{v_{0,wi}} = \frac{\Delta \mathbf{p}_{-1,wi}}{\Delta t_{-1}} \quad (8)$$

where $\Delta\mathbf{p}_{-1,wi}$ is the position change over the preceeding segment of $N$ IMU raw samples and $\Delta\mathbf{t}_{-1}$ the corresponding time. Although calculation of ground truth velocities can suffer from position jitter in the MoCap data and resulting numerical issues when evaluating over a small time window, we believe that these effects can be neglected here due to the high rate of actual MoCap data (360 Hz) in combination with the lower rate of calculated initial velocities ($\sim$ 100 Hz). In addition, the interpolation of ground truth position data as described above also smoothed out some of the position jitter that could otherwise have led to larger errors in the estimated initial velocities. Nonetheless, to exclude the possibility that systematic errors in calculation of initial velocities could have been learnt by the networks, thus providing an unfair advantage, we also compared the performance of the networks to the classical IMU propagation by starting from a known initial position at zero velocity (see Sec. V). Training and evaluation of networks was done in Pytorch (1.9.0).

## V. RESULTS & DISCUSSION

The performance of the models was compared in terms of the mean-square error of the position to standard IMU propagation by a classical filter. For the networks, $M$ predicted, pre-processed IMU samples were integrated to propagate the initial state. A constant bias of zero was assumed. For standard IMU propagation, $N \times M$ raw IMU samples were integrated. Here, a constant bias was taken into account. The biases were determined as described above to be $\mathbf{b_a} = [-0.1161, -0.1581, -0.2281]\ m/s^2$ for the accelerometer and $\mathbf{b}_\omega = [0.0018, 0.0225, -0.0259]\ rad/s$ for the gyro.

An important metric to assess the performance of the IMU propagation is the drift that is accumulated in between subsequent pose updates. A large drift leads to a correspondingly large correction upon the pose update which can result in undesirable large control actions and sudden trajectory changes in closed-loop flight. Depending on the application, a drift of a few centimeters can already lead to critical behavior. We assessed the performance of the models on a validation trajectory, shown in Fig. 4. In order to compare the different models and the standard IMU propagation, random segments of length $M \times N$ were extracted from the trajectory. For each segment, the predicted change in position from the start of the segment to the end of the segment (either based on integration of raw IMU samples or on integration of predicted, pre-processed IMU samples) was compared to the ground-truth information provided by the MoCap system.

The results of the comparison for different numbers of IMU sequences $M$ is shown in Fig. 5. The mean of the

MSE for different number of sequences $M$ ($N$ was kept constant at $N = 8$) is shown as solid line, while the shaded areas indicate the standard deviations. The inset shows the mean MSE for the shorter propagation times (= small $M$). Note that no update steps are performed in the classical filter. Only IMU propagation is considered. This is why the error keeps increasing over time. As can be seen, the LSTM based architecture outperforms both the Transformer encoder based architecture and the standard IMU propagation even for long propagation times. This indicates that an LSTM-based pre-processing of IMU data could handle lower update rates when included in a classical filter framework. While the mean of the MSE of the Transformer encoder architecture is also lower than that of standard IMU propagation (albeit to a lesser extent than with the LSTM), their standard deviations overlap even at long propagation times indicating that the Transformer encoder architecture is less able to detect the predictive features in the IMU samples provided. The inferior performance of the Transformer model is surprising given that it has outperformed LSTM based models in other tasks. One possible explanation is that the Transformer model requires more data for training to be able to reach optimal predictive capability.

To rule out the possibility that the networks learnt to include any systematic errors in the computation of the initial velocities (see Sec.IV-D) which would lead to an unfair advantage over the standard IMU propagation, we also evaluated the cumulative position MSE starting from a well defined position with zero velocities. To that end, the exact start of the take-off maneuver of the validation trajectory was determined, and the IMU samples were extracted from this time onward. The results can be seen in Fig. 6 where the MSE for a propagation of up to 4000 raw IMU samples (corresponding to $M = 500$ pre-processed IMU samples) is shown. Note that since we are following one trajectory in a sequential manner, no standard deviation can be computed and shown in this case.

As can be seen, the behavior is similar to the results obtained from evaluating random sequences: the LSTM architecture again outperforms both the Transformer encoder based architecture as well as the standard IMU propagation, while the Transfomer encoder network also performs better than the IMU propagation initially but then its error increases at a faster rate. Thus, estimating the initial velocities and injecting them in the state for training and testing does not seem to lead to bias the results towards the networks.

## VI. CONCLUSIONS

We have presented a novel approach to improve 6-DoF state propagation by pre-processing the IMU data with the help of neural networks. Contrary to other approaches, we designed our architectures to working with minimal temporal history of IMU data while at the same time allowing for more inference times of the networks and higher state propagation rates. This is achieved by training the networks to down-sample sequences of inertial data provided by a high-rate IMU. The results show that the LSTM based architecture outperforms both the Transformer encoder based architecture as well as standard IMU propagation for various propagation lengths, indicating that this network may be particularly suited to improve state estimation in the presence of lower update rates (e.g. caused by transient lack of sensor readings or failures). Future research will focus on porting these models to the embedded hardware shown in Fig. 3 for closed-loop flight and a more detailed performance analysis, also with respect to carefully balancing the rotational and positional loss terms.

## REFERENCES

[1] S. Weiss, M. W. Achtelik, S. Lynen, M. Chli, and R. Siegwart, "Real-time onboard visual-inertial state estimation and self-calibration of MAVs in unknown environments," in *2012 IEEE International Conference on Robotics and Automation*. IEEE, may 2012.

[2] A. Hardt-Stremayr and S. Weiss, "Monocular visual-inertial odometry in low-textured environments with smooth gradients: A fully dense direct filtering approach," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2020.

[3] E. Allak, A. Hardt-Stremayr, and S. Weiss, "Key-frame strategy during fast image-scale changes and zero motion in VIO without persistent features," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, oct 2018.

[4] C. Brommer, R. Jung, J. Steinbrener, and S. Weiss, "MaRS: A modular and robust sensor-fusion framework," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 359–366, apr 2021.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2017.

[6] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, nov 1997.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, p. 6000–6010.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding."

[9] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, "Transformer networks for trajectory forecasting," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, jan 2021.

[10] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE, apr 2007.

[11] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, aug 2018.

[12] M. Scheiber, J. Delaune, R. Brockers, and S. Weiss, "Visual-inertial on-board throw-and-go initialization for micro air vehicles," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, nov 2019.

[13] E. Allak, R. Jung, and S. Weiss, "Covariance pre-integration for delayed measurements in multi sensor fusion," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019.

[14] M. Brossard, S. Bonnabel, and A. Barrau, "Denoising IMU gyroscopes with deep learning for open-loop attitude estimation," *IEEE Robotics and Automation Letters*, pp. 1–1, 2020.

[15] D. Weber, C. Guhmann, and T. Seel, "Neural networks versus conventional filters for inertial-sensor-based attitude estimation," in *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*. IEEE, jul 2020.

[16] D. Weber, C. Gühmann, and T. Seel, "Riann – a robust neural network outperforms attitude estimation filters."

[17] S. Sun, D. Melamed, and K. Kitani, "Idol: Inertial deep orientation-estimation and localization."

[18] L. Hu, Y. Tang, Z. Zhou, and W. Pan, "Reinforcement learning for orientation estimation using inertial sensors with performance guarantee."

[19] Y. Liu, Y. Zhou, and X. Li, “Attitude estimation of unmanned aerial vehicle based on LSTM neural network,” in *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE, jul 2018.

[20] M. Brossard, A. Barrau, and S. Bonnabel, “AI-IMU dead-reckoning,” *IEEE Transactions on Intelligent Vehicles*, pp. 1–1, 2020.

[21] Z. Zou, T. Huang, L. Ye, and K. Song, “CNN based adaptive kalman filter in high-dynamic condition for low-cost navigation system on highspeed UAV,” in *2020 5th Asia-Pacific Conference on Intelligent Robot Systems (ACIRS)*. IEEE, jul 2020.

[22] F. Wu, H. Luo, H. Jia, F. Zhao, Y. Xiao, and X. Gao, “Predicting the noise covariance with a multitask learning model for kalman filter-based GNSS/INS integrated navigation,” *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–13, 2021.

[23] M. Zmitri, H. Fourati, and C. Prieur, “Inertial velocity estimation for indoor navigation through magnetic gradient-based EKF and LSTM learning model,” in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, oct 2020.

[24] J. N. Wong, D. J. Yoon, A. P. Schoellig, and T. D. Barfoot, “Variational inference with parameter learning applied to vehicle trajectory estimation,” *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 5291–5298, oct 2020.

[25] M. Zhang, M. Zhang, Y. Chen, and M. Li, “Imu data processing for inertial aided navigation: A recurrent neural network based approach.”

[26] S. Weiss, D. Scaramuzza, and R. Siegwart, “Monocular-SLAM-based navigation for autonomous micro helicopters in GPS-denied environments,” *Journal of Field Robotics*, vol. 28, no. 6, pp. 854–874, oct 2011.

[27] P. Furgale, J. Rehder, and R. Siegwart, “Unified temporal and spatial calibration for multi-sensor systems,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2013, pp. 1280–1286.

[28] J. Solà, “Quaternion kinematics for the error-state kalman filter.”