2018 IEEE 15th International Conference on Wearable and Implantable Body Sensor Networks (BSN)
4-7 March 2018
Las Vegas, Nevada, USA

Packet Loss Concealment with Recurrent Neural Networks for Wireless Inertial Pose Tracking

Xuesu Xiao
Department of Computer Science and Engineering
Texas A&M University, College Station TX 77843
Email: xiaoxuesu@tamu.edu

Shuayb Zarar
Microsoft AI and Research
Redmond WA 98052
Email: shuayb@microsoft.com

***Abstract*— Inertial sensing is a technology that enables motion capture outside of well-defined studio environments. Yet, there are several hurdles that have to be overcome in order to achieve a high-quality user experience. Among them is enabling robust wireless communication. Thanks to strict requirements on throughput and far-field operation along with existing issues of occlusion and client interference, packet-loss rates in wireless inertial-sensing systems can amplify pose-tracking errors by as much as 39%. In this paper, we develop a new type of sequence-predictors based on long short-term memory neural networks that can be used to significantly conceal packet losses for inertial pose-tracking. To lower computational overheads, we systematically exploit spatio-temporal correlations of data and distribute sensor loads among multiple predictors. Through experiments conducted with 3.5 hrs. of high-frequency inertial motion-capture data, we demonstrate that our approach is able to fully conceal packet losses at rates of up to 20%.**

I. Introduction

Human pose-tracking, also known as motion capture, is a process that requires continuous estimation of joint angles in 3-dimensional space. Traditionally, this task has been accomplished with the use of cameras that directly detect and track limbs or reflective markers that are attached to the body [1]. Unfortunately, these methods restrict subject movement to within the camera's field-of-view. Non-optical techniques such as those based on inertial sensing provide us with an alternative that has no such restrictions [2]. However, to achieve complete mobility, inertial pose-tracking systems need to be operated wirelessly [3].

Existing wireless inertial pose-tracking systems face many challenges. An important one among them is *data corruption* that occurs due to factors such as channel interference, large operating distances and limb occlusions. Typically, data corruptions manifest as packet losses in the network. Fig. 1 shows the measured packet loss rate (PLR) for a WiFi network at various throughput levels. In this test, we utilized user datagram protocol (UDP) with single socket bindings and a max. allowed stack buffer limit of 1365 Bytes. For comparison, 18 IMUs within a wireless inertial pose-tracking system, transmitting 4 B of single-precision measurements *via* 9 channels each at 30 Hz, require a throughput of ∼150 kbps. At this rate, we see from the figure that data quality degrades significantly when distances from the router are greater than about 60 m; PLR steadily increases to 10% at 90 m. In fact, with an increase in the client or sensor density, these numbers can easily get worse.

At a system level, as PLRs increase, data fidelity suffers leading to jitter in pose tracking. Furthermore, effective operating distances get restricted. In both cases, quality of pose-analysis suffers making it unsuitable for applications that require robust real-time tracking over long distances such as monitoring everyday gait and on-field sports performance. In this paper, we tackle the increase in PLRs by exploiting the temporal relationships in sensor data from inertial motion units (IMUs). Specifically, we develop machine-learning models that learn to predict sequential dependencies in IMU data and fill in missing packets at the receiver enabling a smooth tracking performance. Following are the key contributions that we make:

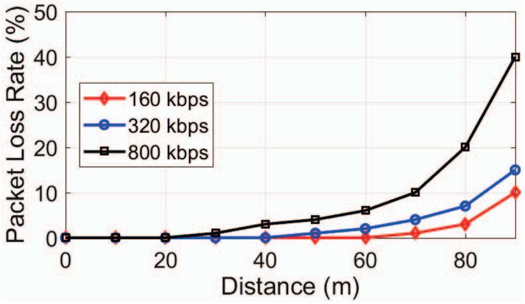


Fig. 1 Measured PLR at different WiFi UDP throughput levels and distances between sensors and the router. Basic inertial pose-tracking in interference-free environments requires a throughput of 150 kbps. Consequently, reliability of data in such a system suffers beyond 60 m of operating distance.

- For the first time, we propose a methodology to exploit recurrent neural networks (RNNs) as sequence predictors to conceal packet losses in wireless networks.
- We apply our methodology to the problem of inertial pose tracking that comprises sensor data with natural temporal dependencies suitable for RNN modeling.
- To lower computational costs of packet-loss concealment (PLC), we develop a technique for vector-RNN processing based on clustered sensor channels.
- We demonstrate that our approach can fully conceal PLRs up to 20%, with modest computational costs.

The rest of the paper is organized as follows. In Sec. II, we present related work on PLC along with background on existing PLR models and long short-term memory (LSTM) networks, which are a specific type of RNN architecture. In Sec. III, we present our proposed methodology of PLC with different neural-network architectures including vectorized learning with sub-clusters of sensor channels. In Sec. IV, we describe our experimental framework and present results along with justifications for different LSTM configurations that we choose. Finally, we conclude in Sec. V.

978-1-5386-1109-8/18/$31.00 ©2018 IEEE

II. Background and Related Work

In this section, we provide an overview of existing methods for PLC along with a mathematical background on LSTMs and PLR models that are widely used in the literature.

A. Existing Methods to Conceal Packet Losses

The issue of packet losses in wireless networks is extensively addressed in communication theory. Primarily, specific routing and medium-access protocols for sensor networks have been developed based on analyses of packet-delivery performance in dense indoor and outdoor wireless environments [4]–[6]. Typically, these approaches achieve PLC within the network-layer by efficient channel coding or re-transmission of data. On the source-coding side, existing techniques rely on dual-channel communication or compressed sensing [7]–[9]. Our approach is closely related to these latter ones. In contrast, however, we employ data-driven methods that treat lost packets as samples within time-series signals. Thus, we avoid communication overheads. Consequently, our proposed techniques can also be applied when factors other than packet losses degrade data quality, such as motion artifacts in free-mode body sensor networks [3], [10].

B. Modeling Packet Loss Rates

PLR in a wireless network can be modeled *via* probabilistic techniques. We specifically focus on UDP, which is desirable for real-time performance in pose-tracking systems. For this protocol, the Gilbert Model is a simple and effective PLR model that is widely used in the literature [11]. It relies on the fact that the probability of losing a contiguous sequence of k packets decreases geometrically with increasing k [12]. It is modeled as a second-order Markov chain with a single random variable X, where $X = 1$ and $X = 0$ represent loss and no-loss of a packet, respectively. For this model, with state probabilities shown in Fig. 2, the matrix of transition probabilities can be expressed as follows:

$$\begin{bmatrix} 1-p & q \\ p & 1-q \end{bmatrix} \begin{bmatrix} P(\text{X=0}) \\ P(\text{X=1}) \end{bmatrix} = \begin{bmatrix} P(\text{X=0}) \\ P(\text{X=1}) \end{bmatrix}.$$

Thus, the conditional and unconditional probabilities of packet loss are given by the following equations:

$$P(X = 1) = \frac{p}{p+q} \quad \text{and} \quad P(X = 1 \mid X = 1) = 1 - q,$$

respectively. Furthermore, since the probability of packet loss only depends on the previous state, the probability of having a loss episode of length k (*i.e.*, P_k), is given by the following:

$$P_k = (1-q)^{k-1} p.$$

The average PLR is $p/p+q$. When q equals $1-p$, this model reduces to a Bernoulli model that is memory less and does not fully characterize burst packet-loss behavior.

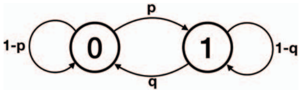


Fig. 2 Gilbert PLR model is based on a 2-stage Markov chain and captures burst behavior. If $p = 1-q$, it reduces to the much simpler Bernoulli model.

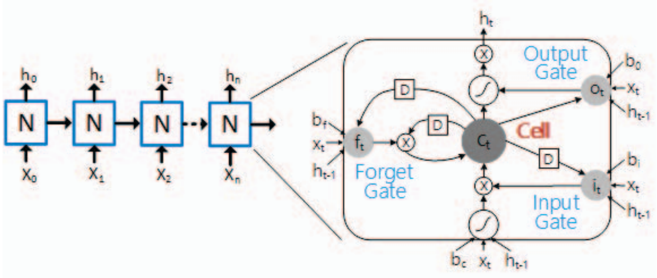


Fig. 3 Unrolled LSTM network with n time steps or hidden units. Its internal memory cell can be controlled *via* the input and forget gates.

C. State Prediction with LSTM Networks

LSTM networks comprise individual units that make use of recurrent connections. Thus, the weighted activation of an LSTM unit is fed back to itself with a delay, which provides it with a memory (hidden value) of past activations allowing the network to learn temporal dynamics in sequential data. Fig. 3 shows an unrolled LSTM network. Each unit comprises three simple gates or neurons: input, output and forget. The output of the LSTM unit is computed as follows:

$$\begin{aligned} c_t &= f_t * c_{t-1} + i_t * tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\ h_t &= o_t * tanh(c_t) \end{aligned} \tag{1}$$

where at time instant t, the outputs of the memory cell and the input, output and forget gates are denoted by c_t, i_t, o_t and f_t, respectively. Furthermore, x_t and h_t are the input and outputs of the LSTM unit, respectively, while b_c is a biasing constant and W_{ic}'s represent weights between the cell and gate i. Next, we show how to exploit the temporal modeling capacity of LSTMs by training them to predict future IMU data in inertial pose tracking.

III. Proposed Approach to Conceal Packet Losses

In this section, we present details on transforming PLC into a sequence-prediction problem. We also demonstrate how to cluster correlated sensor channels to vectorize the PLC computation process.

A. Data-driven LSTM Models for PLC

Once trained with labeled data, an LSTM network can be used to predict 1-dimensional sequences. In order to train it, we utilize the previous $n-1$ data samples, *i.e.*, $x_1, x_2, \ldots x_{n-1}$, and constrain the n^{th} hidden state to be the expected sequence output x_n. With this constraint in place, we minimize gradients over a large number of training examples and back propagate errors to update the LSTM network weights. Thus, we map k individual IMU channels to separate LSTM networks. In this case, n is a hyper parameter that we optimize to achieve the most accurate prediction for all k channels. A summary of our technique is presented in the upper half of the block diagram shown in Fig. 4. Once trained, the network is able to utilize $n-1$ previous samples to predict the n^{th} sample. Thus, if the n^{th} sample is missing because of a lost or corrupted packet, the LSTM fills it in with an estimate of the sample and conceals this defect. In order to maintain consistent internal state of the LSTM units, we need to continually operate all k networks using a sliding window of $n-1$ samples to separately predict x_n's for all k channels. This is not the case if the network is stateless, when we can simply utilize the instantaneous set of $n-1$ previous samples to predict the n^{th} sample.

B. Clustering Sensor Channels for Vectorized Prediction

The model proposed above requires k LSTMs to be trained separately for k IMU channels. Thus, this approach does not fully exploit the inter-relationships between IMU data channels and introduces potential computational overheads at training and inference time. In this section, we propose a methodology to train LSTM networks that simultaneously predict multiple input channels, naturally exploiting temporal relationships in IMU data. Thus, we implicitly enable vectorized-inference utilizing subsets of k sensor channels at a time. Our methodology lowers the number of overall LSTM networks required and even speeds up evaluation on single-core processors that are extremely resource constrained. Unfortunately, experiments show that training one LSTM network for all k channels does not achieve convergence. Therefore, we propose to identify reasonable clusters of sensor channels based on signal correlations.

Our clustering approach is illustrated at the bottom half of the block diagram in Fig. 4 and Algorithm 1. Specifically, since each IMU has 9 channels, we compute the sum of channel-wise correlation coefficients for all s sensors in the pose-tracking system. Thus, between any two sensors i and j, we denote the total correlation coefficient value across all 9 channels by c_{ij}, which lies in the range [0,9]. Subsequently, we perform k-means clustering on these pair-wise sensor-level total correlation values to identify m clusters. To make group index assignments to individual sensors, we consider the frequency of occurrence of each sensor in the m clusters, and choose the cluster index that has the highest occurrence of pair-wise total correlations from a particular sensor. Finally, we train LSTM networks per individual group indices to predict all channels in the group together. Compared to few other potential grouping heuristics, we empirically verify that our approach achieves the best convergence during training time. We posit that this behavior is due to the high levels of correlation between sensor channels that are exploited in individual groups. Next, we present results that demonstrate the benefits of our approach.

IV. Experimental Results

In this section, we present details on the data, pose-tracking algorithm and metrics used for evaluation. We demonstrate that sequence prediction with LSTMs is able to significantly conceal packet losses for wireless inertial pose tracking.

A. Evaluation Framework

An overview of our evaluation framework is shown in Fig. 5. It comprises the following key components: (a) kinematics-based algorithm for pose tracking, (b) data and packet-loss injection, (c) LSTM sequence prediction and (d) network clustering architectures. Next, we present details on each of these components.

Algorithm 1 Correlation-based LSTM Input Clustering

```
Input: $x_{pq}[n]$, $p \in [1, s]$ sensors, $q \in [1, 9]$, # clusters $m$
Output: LSTM sensor groups $\{g_1, g_2, \ldots, g_m\}$
initialize Correlation matrix $\mathbf{C} = \{c_{ij}\}, i, j \in [1, s]$
for $(x_{iq}, x_{jq}) = 1$ to $^sC_2$ do        // Over all sensor pairs
    $c_{ij} = \sum_{q=1}^{9} \rho(x_{iq}, x_{jq})$   // Sum correlation coefficients
end for
Cluster centroids, $\sigma_k \leftarrow$ k-means$[c_{ij}]$,  $k \in [1, m]$
for p =1 to s do                     // Over all sensor indices
    Group index of $x_{pq} \leftarrow$ cluster $k$ where $x_{pq}$ appears
    most frequently
end for
Group $\{g_t\}$ = set of sensors with group index $t, t \in [1, m]$
```

Kinematics-based pose tracking. At the center of the framework is an algorithm that we used to estimate the joint angles. The algorithm is based on kinematic methods that are well-known in the literature [3], [13], [14]. Essentially, we rely on a homogeneous transformation of points in 3D space. For instance, any point can be transformed from frame X to frame Y *via* the rotation matrix R_X^Y. Suppose B_i, S_i and G represent coordinate frames associated with the body segment i, corresponding sensor on the body segment i, and the global reference (aligned with the Earth's magnetic field) at calibration time, respectively. Further suppose B_i' and S_i' represent the body and sensor frames after arbitrary motion. We can express any imaginary point P *via* a transformation of coordinate systems between body segments i and j as follows:

$$R_{S'_i}^{G} R_{B'_i}^{S'_i} R_{B'_j}^{B'_i} P = R_{S'_j}^{G} R_{B'_j}^{S'_j} P \tag{2}$$

By simplifying this expression with matrix inversion, we get:

$$R_{B'_j}^{B'_i} = R_{S'_i}^{B'_i} {R_{S'_i}^{G}}^{-1} R_{S'_j}^{G} {R_{S'_j}^{B'_j}}^{-1} \tag{3}$$

$R_{S'_i}^{G}$ and $R_{S'_j}^{G}$ are the outcomes of the IMU sensor-fusion algorithm computed dynamically [15]. $R_{S'_i}^{B'_i}$ and $R_{S'_j}^{B'_j}$ represent displacement between sensors and body parts. They are obtained during calibration and assumed to remain the same through the course of motion. Thus, by measuring $R_{S'_i}^{G}$ and $R_{S'_j}^{G}$, we are able to compute the joint angle $R_{B'_j}^{B'_i}$ between body parts i and j.

Data and packet-loss injection. We leveraged the Microsoft Inertial Motion Capture (MIMC'17) Dataset, which comprises synchronized recordings from multiple IMUs and infrared (IR) sensors sampled at 30 Hz available as packets delivered over a WiFi link that uses UDP [16]. The locations of the sensors on the body are shown in Fig. 4. In this work, we ignored the IR readings and considered only 38 IMUs that covered 24 body segments. We processed the IMU sensor data to estimate 12 joint angles required for pose tracking. By fusing information from 2 calibrated Kinect sensors, the

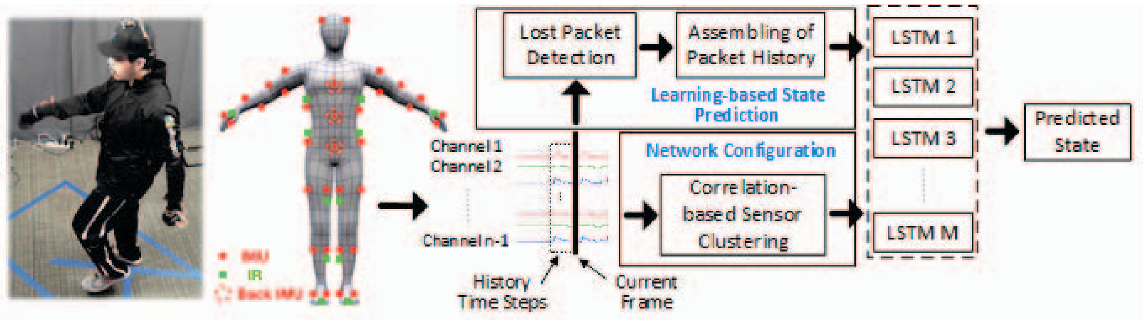


Fig. 4 Overview of the proposed approach. In the basic model, we utilize separate LSTM networks for PLC in individual sensor channels (top of figure). To speed up evaluation with vectorized processing, we exploit sensor correlations and optimize the network configuration (bottom of figure).

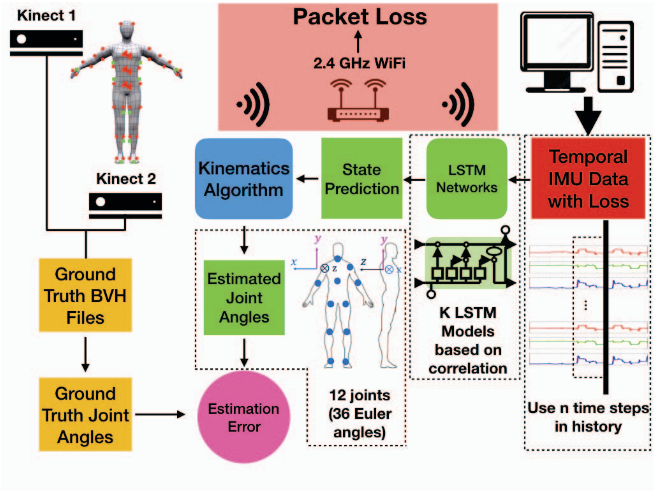


Fig. 5 The proposed experimental framework comprises (a) kinematics-based algorithm for pose tracking, (b) data and packet-loss injection, (c) LSTM sequence prediction and (d) network clustering architectures.

MIMIC'17 dataset also provides us ground-truth joint-angle information in the BioVision (BVH) format. As shown in Fig. 5, we utilized this information to compute baseline values for the errors in the joint-angle estimates produced by the kinematics algorithm described above.

To obtain wireless inertial motion-tracking data that included lost packets at different rates, we invoked the Bernoulli and Gilbert models described in Sec. II-B. Thus, we injected packet-loss errors in the clean IMU data provided by MIMIC'17. For the unconcealed PLR case, we replaced missing packets with the previously received packets. Processing this data with the kinematics algorithm gave us the degraded performance level for pose tracking.

LSTM sequence prediction. We cleaned up the data to conceal packet losses with our state-prediction and clustering algorithms developed in Python. We then processed it with the same kinematics-based algorithm to get an updated estimate of the the joint-angle information. We developed our LSTM models in Theano with Keras APIs. We implemented the proposed models on a PC with 16 GB DDR4 RAM, 3.6 GHz, 16 core 2x Intel Xeon CPUs, and an Nvidia Titan X (Pascal) GPU with 12 GB DDR5 RAM and 3584 Cuda cores running at 1.5 GHz.

Network configurations. We evaluated the efficacy of PLC with our estimation algorithms based on 11 different configurations. They are LSTMs predicting the following:

TABLE I Signal-level MAE *vs.* PLR rates and models.

PLR \ Model	Bernoulli			Gilbert		
	Accel.	Gyro.	Mag.	Accel.	Gyro.	Mag.
5%	0.035	0.040	0.568	0.040	0.048	0.507
10%	0.086	0.099	1.216	0.094	0.101	1.264
20%	0.231	0.269	2.605	0.218	0.252	2.606
30%	0.411	0.480	3.872	0.345	0.392	3.816

TABLE II Mean pose-tracking errors *vs.* PLR rates and models.

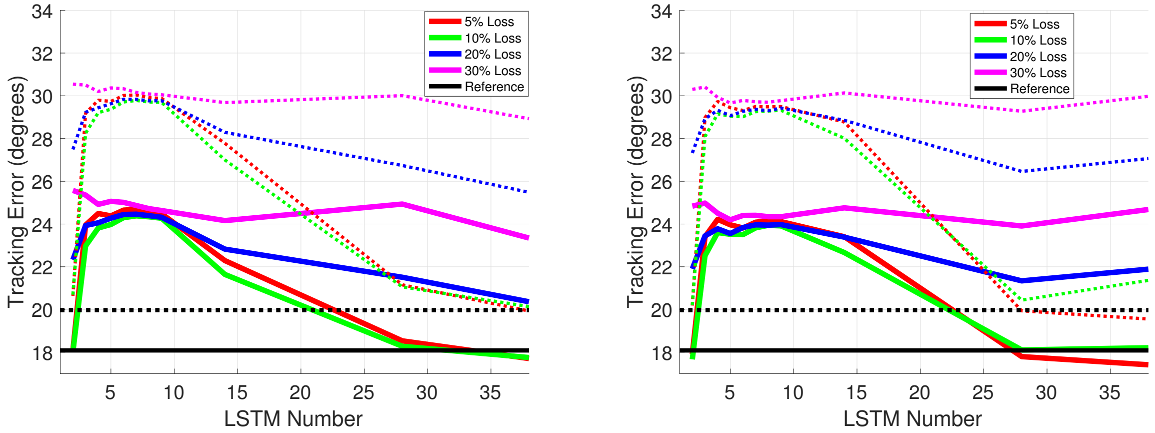
Model \ PLR	5%	10%	20%	30%	No PLR
Bernoulli	17.7°	17.8°	20.4°	23.3°	18.1°
Gilbert	17.4°	17.7°	21.3°	23.9°	

- 2-9 sensor groups using k-means clustering on correlated channels described in Algorithm. 1.
- 14 sensor groups divided by different connected segments of the human body such as feet, arms, torso *etc.*
- 28 groups that further divide the above 14 based on locations on body segments (front, rear, inside or outside).
- 38 sensor groups, each containing only one IMU

B. Concealment Performance

For brevity, we present results that limit the Gilbert burst-parameter q to 0.8. Thus, the loss-transition probability p is controlled solely by the loss rate r and is computed as follows: $p = r \cdot q/(1-r)$. We also restrict evaluation to PLRs of 5, 10, 20 and 30%. At inference time, lost packets are predicted by the trained LSTM models that process different sets of IMU data channels, depending on the clustering mechanisms discussed above. We report results in the form of two metrics: (a) mean-absolute error (MAE) between the LSTM predicted and ground-truth raw IMU sensor data and (b) *end-to-end tracking error* reported as the mean norm of the estimation error in the 3-D Euclidean angle across 12 body joints when compared to the ground-truth optical tracking system with 2-fused Kinect sensors.

Table I shows the best MAE values achieved for the two PLR models at different rates and sensor types. As observed from the table, our approach enables us to reconstruct IMU data quite accurately. Table II shows the impact of this prediction on the mean value of the end-to-end tracking error with the use of the best LSTM configurations. We observe that for PLRs under 10%, in the average case, our best



(a) Bernoulli PLR Model

(b) Gilbert PLR Model

Fig. 6 Our PLC approach reduces tracking errors to the levels of non-lossy data when PLRs are <10%. Mean tracking errors of all joint angles are plotted as solid lines, while standard deviations are shown as dashed lines. The two horizontal lines indicate tracking performance using non-lossy data.

LSTM configurations marginally outperform non-lossy data, potentially due to smoother prediction of the IMU values.

Fig. 6 shows the mean and variance of tracking errors achieved at different clustering levels. The reference tracking-error achieved by the loss-free IMU data in MIMC'17 is also shown as a solid horizontal line. From the figure, we observe that our clustering approach allows us to speed up LSTM processing because of vectorized operations. For sequential single-threaded processors, the speed up factors achieved due to vectorized processing are inversely proportional to the number of LSTMs utilized. This is because of explicitly parallel execution that is made possible in Eq. (1) when x_t and other associated parameters can be processed as vectors. Furthermore, at 10% PLR, the use of 2, 28 or 38 LSTMs could help recover similar tracking accuracy as the non-lossy data for both PLR models. In fact, by processing with just 2 LSTMs, we are able to achieve a reasonable tracking accuracy, while the error starts to increase from 3 groups onwards and plateaus around 6-8 groups. At the peak value, tracking errors are amplified by up to 39% compared to the baseline. This behavior is consistent at PLRs below 20%. However, at 20% PLR, careful configuration of the LSTM architecture becomes necessary to lower tracking errors. However, at more than 30% PLR, we were not able to improve the tracking accuracy significantly.

V. Conclusions

In emerging applications of wireless inertial pose tracking, real-time jitter-free performance is crucial for a good user experience. Loss of packets in a wireless network presents a hurdle to achieving this objective. In this paper, we explored the possibility of utilizing the sequence prediction power of LSTM networks to conceal these packet losses. We observed that in resource-constrained scenarios, where these algorithms are expected to run, utilizing an LSTM for individual IMU data channels is not scalable. Thus, we proposed an approach to cluster sensor channels, enabling us to run our packet-loss concealment framework in a vectorized manner with substantially lower execution times even on single-threaded, single core processors.

References

[1] M. Windolf, N. Götzen, and M. Morlock, "Systematic accuracy and precision analysis of video motion capturing systems-exemplified on the Vicon-460 system," *J. Biomechanics*, vol. 41, no. 12, pp. 2776–2780, Jan. 2008.

[2] D. Vlasic *et. al.*, "Practical motion capture in everyday surroundings," in *ACM Trans. Graphics*, vol. 26, no. 3. ACM, Jul. 2007, p. 35.

[3] X. Xiao and S. Zarar, "Machine learning for placement-insensitive inertial motion capture," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[4] J. Zhao and R. Govindan, "Understanding packet delivery performance in dense wireless sensor networks," in *Proc. Int. Conf. Embedded Networked Sensor Systems*, 2003, pp. 1–13.

[5] J. G. Kim and M. M. Krunz, "Bandwidth allocation in wireless networks with guaranteed packet-loss performance," *IEEE/ACM Trans. Networking (TON)*, vol. 8, no. 3, pp. 337–349, Feb. 2000.

[6] J. Tang and X. Zhang, "Quality-of-service driven power and rate adaptation over wireless links," *IEEE Trans. Wireless Communications*, vol. 6, no. 8, 2007.

[7] C. Chigan and V. Oberoi, "Providing qos in ubiquitous telemedicine networks," in *Int. Conf. Pervasive Computing and Communications Wkshp.* IEEE, 2006, pp. 5–pp.

[8] H. Garudadri and P. K. Baheti, "Packet loss mitigation for biomedical signals in healthcare telemetry," in *Int. Conf. Engineering in Medicine and Biology Society*, 2009, pp. 2450–2453.

[9] P. K. Baheti and H. Garudadri, "Heart rate and blood pressure estimation from compressively sensed photoplethysmograph," in *Proc. Int. Conf. Body Area Networks*, 2009, p. 13.

[10] S. Mohammed and I. Tashev, "Unsupervised deep representation learning to remove motion artifacts in free-mode body sensor networks," in *Int. Conf. Body Sensor Networks*, May 2017, pp. 183–188.

[11] G. Haßlinger and O. Hohlfeld, "The gilbert-elliott model for packet loss in real time services on the internet," in *Measuring, Modelling and Evaluation of Computer and Communication Systems (MMB), 2008 14th GI/ITG Conference-*. VDE, 2008, pp. 1–15.

[12] V. Markovski, F. Xue, and L. Trajković, "Simulation and analysis of packet loss in user datagram protocol transfers," *The J. Supercomputing*, vol. 20, no. 2, pp. 175–196, 2001.

[13] J. Favre *et. al.*, "Functional calibration procedure for 3D knee joint angle description using inertial sensors," *J. biomechanics*, vol. 42, no. 14, pp. 2330–2335, Oct. 2009.

[14] G. Pons-Moll *et. al.*, "Outdoor human motion capture using inverse kinematics and von mises-fisher sampling," in *Int. Conf. Computer Vision*, Nov. 2011, pp. 1243–1250.

[15] S. O. Madgwick, A. J. Harrison, and R. Vaidyanathan, "Estimation of IMU and MARG orientation using a gradient descent algorithm," in *Int. Conf. Rehabilitation Robotics*, Jul. 2011, pp. 1–7.

[16] (2017) MSR inertial motion capture dataset. [online] available at https://www.microsoft.com/en-us/download/details.aspx?id=56054.