sensors

MDPI

Article

Capturing Upper Body Kinematics and Localization with Low-Cost Sensors for Rehabilitation Applications

Anik Sarker [1], Don-Roberts Emenonye [2], Aisling Kelliher [3], Thanassis Rikakis [4], R. Michael Buehrer [2] and Alan T. Asbeck [1,*]

[1] Department of Mechanical Engineering, Virginia Tech, Blacksburg, VA 24061, USA; aniks@vt.edu
[2] Department of Electrical & Computer Engineering, Virginia Tech, Blacksburg, VA 24061, USA; donroberts@vt.edu (D.-R.E.); rbuehrer@vt.edu (R.M.B.)
[3] Department of Computer Science, Virginia Tech, Blacksburg, VA 24061, USA; aislingk@vt.edu
[4] Department of Biomedical Engineering, University of Southern California, Los Angeles, CA 90089, USA; rikakis@usc.edu
* Correspondence: aasbeck@vt.edu

Abstract: For upper extremity rehabilitation, quantitative measurements of a person's capabilities during activities of daily living could provide useful information for therapists, including in telemedicine scenarios. Specifically, measurements of a person's upper body kinematics could give information about which arm motions or movement features are in need of additional therapy, and their location within the home could give context to these motions. To that end, we present a new algorithm for identifying a person's location in a region of interest based on a Bluetooth received signal strength (RSS) and present an experimental evaluation of this and a different Bluetooth RSS-based localization algorithm via fingerprinting. We further present algorithms for and experimental results of inferring the complete upper body kinematics based on three standalone inertial measurement unit (IMU) sensors mounted on the wrists and pelvis. Our experimental results for localization find the target location with a mean square error of 1.78 m. Our kinematics reconstruction algorithms gave lower errors with the pelvis sensor mounted on the person's back and with individual calibrations for each test. With three standalone IMUs, the mean angular error for all of the upper body segment orientations was close to 21 degrees, and the estimated elbow and shoulder angles had mean errors of less than 4 degrees.

Keywords: kinematics; inertial sensors; self-supervised learning; sparse sensors; activity recognition; human pose estimation; localization; proximity reporting; Bluetooth beacon; Bluetooth RSS

Citation: Sarker, A.; Emenonye, D.-R.; Kelliher, A.; Rikakis, T.; Buehrer, R.M.; Asbeck, A.T. Capturing Upper Body Kinematics and Localization with Low-Cost Sensors for Rehabilitation Applications. *Sensors* **2022**, *22*, 2300. https://doi.org/10.3390/s22062300

Academic Editor: Andrea Cataldo

Received: 1 February 2022
Accepted: 11 March 2022
Published: 16 March 2022

1. Introduction

1.1. Overview

As the US population ages, there is an increasing need for effective and accessible rehabilitation services for debilitating illnesses and injuries such as stroke and degenerative arthritis [1,2]. Effective rehabilitation requires intensive training and the ability to adapt the training program based on patient progress and therapeutic judgment [3]. Telemedicine and telehealth are gaining prominence as avenues for delivering participatory health and wellness in the home at scale. However, a practical approach to physical rehabilitation in the home is not yet possible due to the challenges in capturing meaningful data about how the patient is progressing in a low-cost, easy-to-use way. For upper extremity rehabilitation for stroke survivors, over 30 low-level movement features need to be tracked as the patient performs functional tasks in order to precisely and computationally characterize movement impairment [4]. In addition, detailed activity documentation during daily life is needed to understand the effect of therapy on functional recovery [5]. Although high-end sensing technologies can provide some of the necessary detailed tracking, these technologies are cumbersome even in the clinic and certainly not yet feasible for the home. Tracking of

movement through marker-based capture or full-body inertial measurement unit (IMU) systems is impractical and often costly [6–10]. Systems such as exoskeletons or other devices that must be worn along the arm can be cumbersome and may lead to low patient compliance [11,12]. Video or depth camera arrays [13,14] may be objectionable for patients and home occupants due to the feeling of being under constant surveillance [15]. Traditionally, accelerometry has been used to give information about a patient's motion in a home environment [16–19], but this provides only coarse measures of patient capability. Some work has also been done in activity recognition in the home [20], or have combined a patient's location in a home environment with estimates of their activity [21–23]; however, quantifying a person's actual arm kinematics may be more useful than activity recognition.

Consequently, there is a need for low-cost but accurate technologies that can accurately capture a patient's functional movements during daily life. With kinematics sensing, a patient's motions can be assessed to monitor progress with rehabilitation. Importantly, contextualizing a person's motions may be important to determine the circumstances in which they do not use their limbs normally or perform compensatory motions. With this information, therapists could determine the best course of action for rehabilitation. In this paper, we propose a system that can capture both the location of a patient within their home and also their upper body kinematics. As seen in Figure 1, this consists of two components: First, a system based on Bluetooth that can localize the patient within the home (Figure 1a). Second, a system that uses a minimal sensor set to infer the complete upper body kinematics (Figure 1b,c). With these, we present the initial steps towards a practical at-home tele-rehabilitation system.

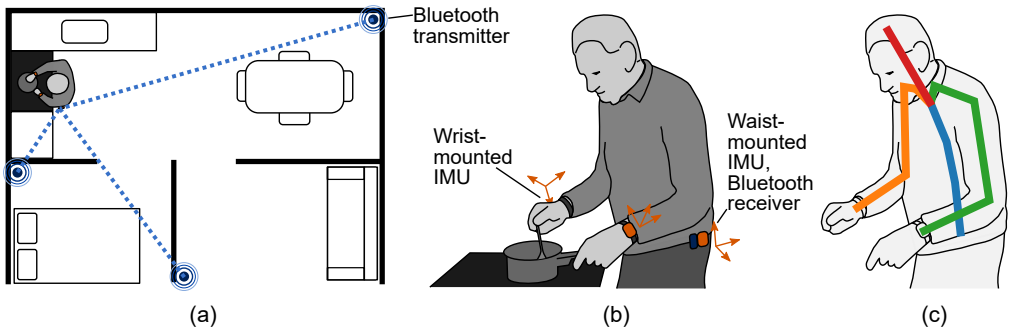


Figure 1. Overview of our strategy for in-home localization and kinematics monitoring. (**a**) In-home localization approach. Bluetooth transmitters are installed around the home, and the received signal strength is monitored at the patient. The localization provides context to the captured activities. (**b**) Minimal sensor set that is unobtrusive during daily life, including IMUs worn on each wrist and the waist, and a Bluetooth receiver worn at the waist. (**c**) Kinematic reconstruction of the torso derived from the worn IMUs.

1.2. Background and Related Work on Localization

Location is a crucial context for determining activity. Although the worldwide Global Positioning System (GPS) can provide sub-centimeter level position accuracy, this capability does not extend to indoor scenarios with an absent line-of-sight path from the target to the GPS satellites [24]. Nonetheless, location awareness will still serve as an enabler for indoor health care systems. Location awareness in indoor systems can be enabled by measuring both wireless propagation characteristics (transmitted by known beacons) and motion-related characteristics like acceleration (through accelerometers) and angular velocity (through gyroscopes). The accurate estimation of these characteristics enables location inference.

There are two schools of thought employed in mapping wireless characteristics to location estimates: (1) model-based approaches and (2) fingerprinting-based approaches. In the former, researchers usually assume a wireless propagation model. Subsequently, data points are collected at reference points (RPs) and are used to fit the assumed propagation

model. Hence, given observed/estimated wireless characteristics, the distance is readily derived from the assumed propagation model. The difficulty in this approach lies in deriving the appropriate propagation model given that wireless propagation is complex and can vary substantially from location to location [25].

In contrast, fingerprinting-based approaches treat wireless signal measurements as signatures observed in space, frequency, and time. During training, these wireless signatures are observed and intelligently associated with particular locations [25]. After deployment or during testing, new wireless characteristics are observed and compared with previous signatures; and through the association learned during training, the new locations are approximated. The two fundamental building blocks of fingerprinting-based approaches are the association algorithm used and the wireless characteristics selected as signatures.

The most commonly used signature in fingerprinting approaches is the received signal strength (RSS). This commonality is because RSS can be easily obtained from wireless receivers found in phones, Raspberry Pis, and computers [26]. In [26–31], an RSS-based fingerprinting database was developed, and the location estimate of a new RSS value was derived as a function of the locations of the k most similar RSS values in the database. In [27], the location estimate of the new RSS value was obtained by simply averaging the location of the k nearest values in the database. In [26], a Spearman criterion for ranking the k-nearest neighbors (k-NN) is provided, and the effects of varying k on the accuracy of the location estimate are investigated. In [28], the authors recognize that the similarity distance used in prior k-NN works incorrectly assumes that similar RSS values translate to similar geometric distances. The authors compensate by proposing a modified feature scaling-based k-NN. In [32], a correlation database (CDS) for fingerprinting is built based on the Okumura-Hata propagation model [33]. In that work, to ensure the transmit power is known, the database is built based on transmissions in the control channel. In [34], by relating the locations where RSS signatures are collected to genetic chromosomes, a genetic algorithm [35] is applied to reduce the size of the fingerprinting database.

In addition, artificial neural networks (ANN) have been proposed as fingerprinting-based association algorithms. In [36], a single hidden layer neural network is trained and used to provide location estimates at test time. The neural network has three input nodes, 16 hidden nodes, and two output nodes. The input nodes correspond to the RSS observed at the target from the three access points, and the output nodes provide 2D location estimates. This neural network design provided an accuracy of 1.75 m. In [37], the previous work is extended to a neural network with multiple layers. In that work, the neural network is divided into a data processing section, a denoising section, and a location estimation section. The neural network input is the RSS from the access points, while its output is the 2D location estimate of the target. More recently, a recurrent neural network has been proposed for location estimation [38]. Authors in that work recognize that RSS values received while a target is on a trajectory will be correlated. With this, a recurrent neural network (RNN) enabled trajectory positioning scheme is developed. Recently, a dynamic model estimation technique has been used for indoor positioning [39]. In [40], a chest-mounted IMU is proposed for indoor positioning. In [41], a systematic review is provided for collaborative indoor positioning techniques. A study on indoor positioning systems in harsh wireless propagation environments is presented in [42]. Finally, an automatic context-based positioning system based on Wi-Fi is presented in [43].

Although RSS-aided positioning has been studied rigorously, the requirements for a wireless positioning system in a smart health context are considerably different. For instance, health care professionals are more interested in localizing the subject to a region of interest (RoI) than localizing to the exact coordinates. Hence, for healthcare systems, it will be more suitable to provide proximity reports. Proximity reports reveal how close a subject is close to a set of anchors. A proximity report can be specified by a binary vector $\mathbf{y} = [1,\ 0,\ 1,\ 0]$ where the i-th element in the vector $\mathbf{y}$ specifies whether the subject is in the vicinity of the i-th reference node. Clearly, the intersection of the respective vicinity confines the subject to a specific unambiguous RoI. One way to generate proximity reports is by

comparing the instantaneous RSS received from different access points to pre-defined RSS thresholds. These thresholds can be derived in a cooperative or non-cooperative fashion. Proximity reports describe the vicinity of the desired subjects without explicitly providing their location estimates. In [44], a campaign is conducted to measure RSS values from different access points at various reference locations. The data collected is used to fit both a linear log-distance model and a Gaussian process regression. Subsequently, the collected data is used to find the optimal threshold for proximity reporting. The selected optimality criterion is the Cramer-Rao bound (CRB). In [45], the work is extended to incorporate multiple thresholds for each reference point. Furthermore, the Barankun bound [46] is used as an optimality criterion. Authors in [47] derive the CRB for a K-level RSS quantization scheme. In that work, it is shown that the lower bound on the MSE for proximity location is 50% higher than the bounds in conventional RSS-based systems. Although these prior works are promising, the optimization thresholds are based on propagation models, which may not represent the wireless environments' actual characteristics. Moreover, most of these works fail to consider the correlation in RSS due to the desired subject trajectory. In order to circumvent the need to assume a model, we propose employing deep neural networks (DNNs) to generate proximity reports. We also propose an RNN to account for the correlation between the current RSS values and previous RSS values. Lastly, as a separate contribution, we validate an already existing algorithm [29] with experimental data. In [29], an improved k-NN algorithm was proposed, but was not validated with real-world data. We test the accuracy of that fingerprinting technique proposed for simulated data with real data.

1.3. Background and Related Work on Motion Inference

A number of prior works have examined the problem of motion inference via sparse sensors, i.e., predicting the joint angles for the entire body by using only a few sensors.

Several works have used IMUs in conjunction with a video camera [48–52] or depth cameras [53,54]. Generally speaking, the fusion of two different sensor technologies is beneficial; however, for our application, it is impractical and privacy-invasive to use cameras inside a home environment. Another work used RFID tags in conjunction with IMUs to provide more information [55].

A number of recent works have used solely IMUs to reconstruct kinematics [56–61]. These have used a variety of approaches for motion inference. Initially, Gaussian processes were used by [56]. Next, ref. [58] used an optimization-based approach, which required knowledge of the person's initial pose and the sensor locations on their body, with impressive results. More recently, neural networks have been used for motion inference [59–61]; these have each used bidirectional long short-term memory (LSTM) neural network architectures [62], where the time history of each sensor provides cues to the current kinematic pose. Both Huang [59] and Yi [61] train their models on the AMASS dataset [63] and the TotalCapture dataset [64] as well as another dataset collected by Huang called DIP-IMU. Their architectures are somewhat similar, but Yi uses a dedicated processing step to estimate ground-foot contacts. Both groups use six IMUs worn on the wrists, lower legs, pelvis, and head, and predict the full-body kinematics.

In our previous work, we used a custom dataset and various numbers of sensors to perform motion inference [60]. The dataset, called the Virginia Tech Natural Motion dataset, contains kinematic recordings of people doing activities of daily living as well as stockers in a warehouse environment. The data was captured with an XSens MVN Link system [65,66] and contains more than 40 h of motion. Using this dataset, we conducted motion inference of both the whole body and the upper body based on 3–6 different body segments. Specifically, we used the XSens-generated orientations and accelerations from each segment to infer the other joints. We note that the orientations and accelerations of the segments are based on the whole-body kinematic reconstruction; thus, their values are somewhat different than if a standalone IMU was placed on each segment. In the present work, we use sparse standalone IMUs to perform kinematics reconstruction. The results

are not as good as gold-standard motion capture, but may be sufficient to understand a patient's motion for rehabilitation.

1.4. Contributions

In this paper, we have several contributions. Overall, we present a new strategy for understanding human motion during activities of daily living with just a few unobtrusive sensors, both determining the location of an individual within their home and estimating their kinematics.

In the area of localization, we present new algorithms based on Bluetooth beacons to reduce the uncertainty of a person's position to an RoI. Our first contribution to both the general area of positioning and in the area of positioning for health care is to develop DNNs for proximity reporting. Similar to existing DNNs for positioning, the neural network tries to learn the nonlinear relationship between the RSS and the target location, but unlike existing DNNs, the neural network does not produce a 2D or 3D location estimate. Instead, the DNN produces a vector that describes the vicinity of the target location. This structure is similar to multilabel classification [67] in machine learning theory, in which a single observed sample can belong to multiple classes. In this paper, we perform simulations to demonstrate the algorithms.

Since healthcare systems use location context to provide recommendations to patients, it is not an absolute necessity to have the exact coordinate of the patients. In this scenario, it is sometimes more important to be able to determine what vicinity the patient is in. Hence, the proposed proximity reporting technique can find application in healthcare systems. However, to have the option of determining the exact coordinates of the patient, our second contribution for positioning estimation is to validate the improved k-NN algorithm proposed in the literature [29] with actual Bluetooth beacons. The localization operation in this scenario is divided into training and test stages. During the training stage, the BLE signals from the beacons are collected using a Raspberry Pi [68]. The Raspberry Pi is synchronized with the Beacons and programmed to time stamp the Beacon data and store their RSS values. These RSS values are used to build a fingerprinting database for positioning. During the testing stage, new RSS values are collected and compared with the RSS values in the database. This comparison is used to provide a location estimate. We demonstrate the algorithm with experimental data in a home environment.

In the area of kinematics estimation, we use standalone IMUs combined with our motion inference algorithms [60] to generate an estimate of upper body kinematics during activities of daily living. While several works have examined inferring kinematics of the entire body using sensors on the arms, legs, and torso or head, we use a reduced sensor set with only off-the-shelf sensors on the wrists and pelvis to infer only the upper body. This sensor set is simple, unobtrusive, and easy to use during daily life, especially for people in need of rehabilitation. We compare the accuracy of upper-body kinematic inference using standalone IMUs to information from the ground truth whole-body kinematics. We present the kinematic inference accuracy for each individual joint in the torso since, in rehabilitation contexts, it is useful to understand which joints need additional attention. We also examine the differences in performance between putting the pelvis sensor on the back of the pelvis (as was done previously) versus the side, a location that is more suitable for long-term wear in the home.

The rest of the paper is organized as follows. In Section 2, we present our algorithms and methods for experimental evaluation of localizing a person in a home environment. In Section 3, we present our algorithms and experimental evaluation methods for inferring the kinematics of the upper body. In Section 4, we present all of our experimental results, and in Section 5, we provide the discussion.

2. Materials and Methods for Positioning

2.1. Overview

In this section, we discuss our localization algorithms and the experimental setup for their evaluation. In the next section, we discuss our kinematics reconstruction algorithms and their experimental evaluation.

2.2. Methods for Localization—Proximity Reporting

In this section, we consider a proximity reporting-based technique for indoor positioning, where RSS received from a set of anchors/beacons is compared to predetermined thresholds to determine the position of a target. Note that the actual coordinates of the target is not provided by the proximity reports, the proximity reports only confines the target to a region of interest (RoI). We consider a simulated environment with a set of U anchors with known locations in a two-dimensional grid. The locations of the U anchors can be defined as:

$$\boldsymbol{U} = \begin{bmatrix} x_1 & x_2 & \cdots & x_U \\ y_1 & y_2 & \cdots & y_U \end{bmatrix}.$$

The goal is to find the position of a target described with the following vector $\boldsymbol{s} = [x, y]^T$. Each anchor has a wireless transmitter with Bluetooth 4.0 capabilities that broadcasts ibeacon packets. Each ibeacon packet contains a unique identifier (UUID) that is unique to the broadcasting transmitter. The anchors broadcast at a sampling frequency of 10 Hz, i.e., a single packet is broadcast every 100 ms. A Bluetooth receiver attached to the target collects and stores the packets. The RSS of the signal from each anchor is also stored along with the associated UUID. This UUID differentiates the packets from different anchors. The received power measured in dBm at the target from the uth anchor can be characterized as:

$$r(d_u) = P_t - \bar{r}(d_u) + X_{\sigma_u}, \tag{1}$$

where P_t (dBm) is the transmit power of the source, $\bar{r}(d_u)$ is propagation loss at a distance d_u, $d_u = \|s - s_u\| = \sqrt{(x - x_u)^2 + (y - y_u)^2}$ and $X_{\sigma_u} \sim \mathcal{N}(0, \sigma_u)$ is a slow-fading term due to shadowing. The propagation loss can be written as:

$$\bar{r}(d_u) = PL_u(d_0) + 10\xi_u \log\left(\frac{d_u}{d_0}\right), \tag{2}$$

where $PL_u(d_0)$ is the path loss measured at a reference distance d_0, and ξ_u is the path loss exponent [25]. Because $PL_u(d_0)$ is deterministic, the equivalent mean RSS can be written as:

$$\begin{aligned} \bar{v}_u &= PL_u(d_0) - \bar{r}(d_u), \\ &= 10\xi_u \log\left(\frac{d_0}{d_u}\right). \end{aligned} \tag{3}$$

Clearly, $\bar{v}_u$ is dependent on the hub/target position $\boldsymbol{s}$ and the random variable specifying the RSS is given as:

$$v_u = \bar{v}_u + X_{\sigma_u}. \tag{4}$$

Due to lognormal random variable, the ith sample from the uth anchor $v_{u,i}$ can be characterized by a lognormal distribution:

$$f_v(v_{u,i}) = \frac{1}{\sqrt{2\pi}\sigma_u} \exp\left(\frac{-(v_{u,i} - \bar{v}_{u,i})^2}{2\sigma_u^2}\right). \tag{5}$$

The model specified by Equations (3)–(5) is used to derive and optimize thresholds in [44,45,69]. However, these thresholds are complex and heavily dependent on the specific environment. To circumvent this challenge, we propose to use a neural network to generate the proximity reports.

2.2.1. Overview of Neural Network

Deep neural networks act as universal function approximators that can learn the complex relationship between an observation and its label. Given an unknown function, f^*, that completely describes the observations and their labels in a dataset $\{\mathbf{S}, \mathbf{y}\}$, a DNN tries to learn a set of parameters $\boldsymbol{\vartheta} = \{\vartheta^1, \vartheta^2, \cdots, \vartheta^Z\}$ that can produce an approximation of f^* as f. Here, Z represents the number of neural network layers. A simple deep learning network usually has no feedback loop, and its operation can be described as:

$$f(\boldsymbol{s}) = f^{(Z)}(\cdots f^{(2)}(f^{(1)}(\boldsymbol{s}))), \tag{6}$$

where f^1, f^2, and f^Z, represent the 1st, 2nd, and Zth layers respectively. The operation of the Zth layer can be completely described as

$$\mathbf{w}^z = f_{\vartheta^z}(\mathbf{w}^{z-1}) = \mathrm{Y}^l(\mathbf{W}^z\mathbf{w}^{z-1} + \boldsymbol{\zeta}^z), \tag{7}$$

where $\mathbf{W}^z$ and $\boldsymbol{\zeta}^z$ describes the weight and bias terms of the Zth layer, $\mathbf{w}^{z-1}$ describes the output of the previous layer, Y^z denotes the activation function of the Zth layer, and $\boldsymbol{\vartheta}^z = \{\mathbf{W}^z, \boldsymbol{\zeta}^z\}$ denotes the parameters of the Zth layer. Clearly, the operation of the neural network layers can be viewed as a linear transformation empowered by the activation function. In this work, we will restrict our choice of activation functions to the popularized ReLu function. Although DNNs are adept at learning the complex relationships between the input and output, they are not structured to learn temporal correlation. This is intuitive because a plain DNN does not contain any feedback loops, as shown in Figure 2. In order to solve this challenge, recurrent neural networks (RNNs) with built-in loops were developed. These loops allow for information to persist from one time step to another. An RNN is designed to learn the temporal among a sequence of inputs.

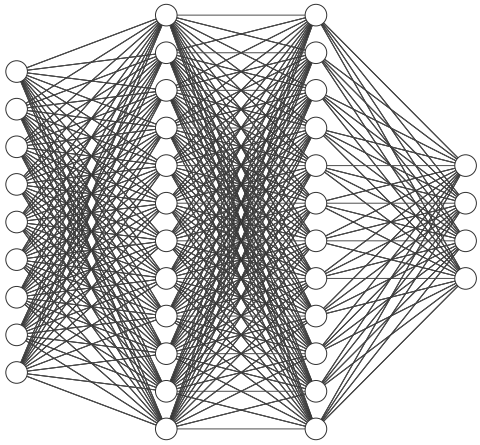


Figure 2. Schematic of a deep neural network with an input layer capable of accepting a nine feature input vector, two hidden layers each with 12 neural network nodes, respectively, and an output layer of 4 nodes.

In Figure 3, the Zth layer of the recurrent neural network accepts as input $\mathbf{w}_t^z$, and produces output $\mathbf{w}_t^{z+1}$. The loop allows for information to be shared across time instances.

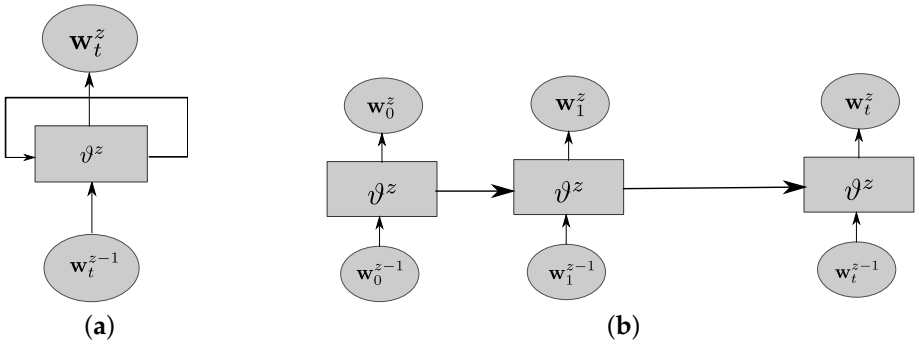


Figure 3. (**a**) A compact representation and (**b**) an unrolled representation of a recurrent neural network.

2.2.2. System Setup and Data Generation for Proximity Reporting

The rest of this sub-section will be focused on developing an indoor positioning system with RSS as the selected wireless propagation characteristic and a recurrent neural network as the selected signature-to-location association function. The output of the neural network is a vector that describes the vicinity of the target. This vector is similar to multi-label classification in image processing [67]. We consider a simulated 50 m by 50 m indoor patient rehabilitation center, which is divided into ten subcenters. An anchor was placed at the middle of each subcenter. Hence, $U = 10$ anchors each equipped with Bluetooth low energy (BLE) beacon transmitters. On entry, the simulated patients were equipped with a mobile hub capable of measuring RSS data from the U anchors. The simulated patients were asked to interact with one another and ensure that they are in motion for a particular time interval. The simulated therapists tasked with rehabilitation offer different instructions on physical activities to the simulated patient in the form of push notifications depending on whether they are in the vicinity of certain anchors. Figure 4a shows the grid with 10 access points.

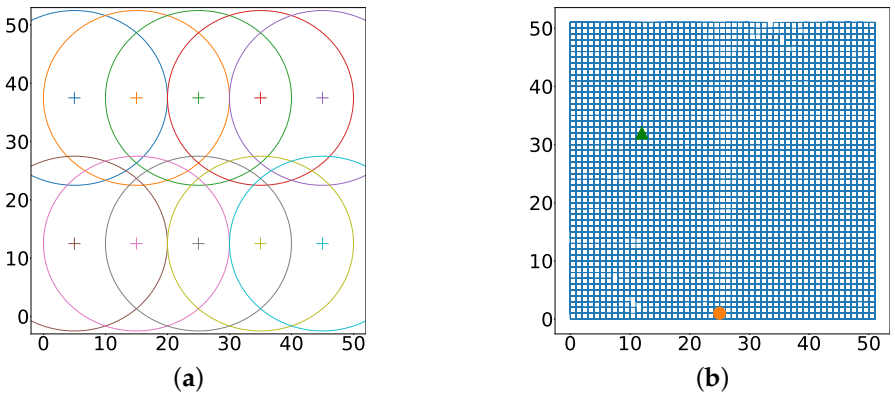


Figure 4. (**a**) Representation of the grid: the vicinity next to an anchor is defined as 15 m from the anchor. (**b**) Simulated patient bounded random walk.

We model the movement of the simulated patient as a bounded random walk from the green arrow to the orange circle. We assume that U RSS values sampled from a log-distance model are received at the patient's hub at every time step. The data received over the time period given by T_{train} is the training data. The log-distance characteristics of each anchor is given in Table A2.

The received RSS values are pre-processed by clipping to ensure that they lie within the range $[-100, -50]$ dBm. The clipping operation can be defined as:

$$\nu_u = \begin{cases} -100 \text{ dBm}, & \text{if } \nu_u < -100 \text{ dBm}, \\ -50 \text{ dBm}, & \text{if } \nu_u > -50 \text{ dBm}, \\ \nu_u, & \text{Otherwise.} \end{cases} \tag{8}$$

2.2.3. Training of Recurrent Neural Network for Proximity Reporting

This section focuses on the training of a long term short memory (LSTM) type of an RNN for proximity reporting. We use a dataset of T_{train} training samples, each with U number of features. The features of the ith training example can be described as $\mathbf{s}_{t,i} = \{\nu_1, \nu_2, \cdots, \nu_U\}$. The dataset is collected offline and each training sample has a label describing the vicinity of the target. Unlike prior works, that use the location estimates as a label, the label is described as a vector $\mathbf{y}$ in which its ith element is specified as:

$$\mathbf{y}^{[i]} = \begin{cases} 1, & \text{if the patient is within the vicinity of the } i\text{th anchor}, \\ 0, & \text{otherwise.} \end{cases}$$

The input vector $\mathbf{s}_{t,i}$ is standardized so all the features lie between 0 and 1. This vector serves as input to the LSTM layers, which has a memory of T_{LSTM} time steps. The neural network is depicted below in Figure 5.

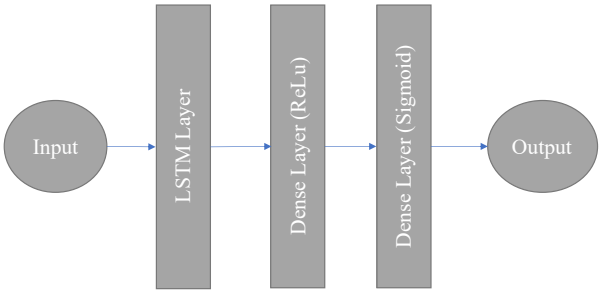


Figure 5. Deployed LSTM neural network for proximity detection.

The estimate of the patient's vicinity, vector $\hat{\mathbf{y}}$ can be written as:

$$\hat{\mathbf{y}}^{[i]} = \begin{cases} 1, & (\mathbf{w}^Z)^{[i]} > 0.5, \\ 0, & \text{otherwise.} \end{cases}$$

The cross entropy loss function used for training can be written as:

$$\mathcal{L}(\boldsymbol{\vartheta}) = -\frac{1}{T_{train}} \sum_{t=1}^{T_{train}} \mathbf{y}_t^T \log(\mathbf{w}^L). \tag{9}$$

With this loss function, and a learning rate, α, the stochastic gradient descent algorithm is used to update the neural network parameters as:

$$\boldsymbol{\vartheta} := \boldsymbol{\vartheta} - \alpha \nabla \mathcal{L}(\boldsymbol{\vartheta}). \tag{10}$$

The training parameters are given in Table 1.

Table 1. Training parameters.

Parameter	Value
Learning Rate, α	0.01
Iterations (epochs)	400
Batch size, T_{train}	500
Training Shadowing variance, σ_s	7.5

2.2.4. Testing Stage for Proximity Reporting

To test the proximity detection system, the simulated patient equipped, as previously described, performs another random walk starting from the green arrow and ending at the orange circle as shown in Figure 6.

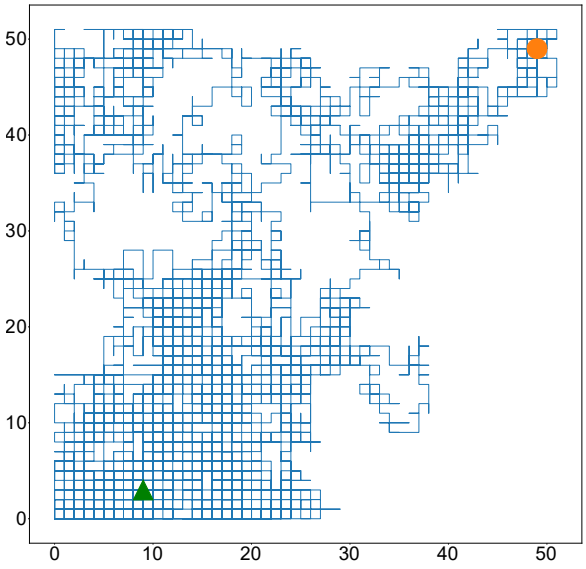


Figure 6. Trajectory used for testing.

To show the accuracy of the recurrent neural network, we define the following performance metrics:

- Proximity accuracy: this specifies the indoor system's ability to detect whether the simulated patient is within a predefined range from the anchor.
- Distance accuracy: this specifies the indoor system's ability to detect when the simulated patient is not within a predefined range from the anchor.
- Overall accuracy: this specifies the indoor system's ability to either place the simulated patient within range from the anchor or to determine the absence of the simulated patient within a certain range from the anchor.

2.3. Localization with Real Data

In this section, we present a localization technique with real-world data. The RSS is normally distributed in dB and related to the target position. Hence, assigning a unique signature, known as an RSS fingerprint, to different locations is possible. The fingerprint at the target can be described as $\boldsymbol{G} = [\nu_1, \nu_2, \cdots, \nu_U]$. The fingerprint is very useful if it varies substantially from one location to another. Fingerprint-based approaches treat RSS as signatures observed in space and time. This fingerprinting operation can be divided into training and testing stages. The training stage involves building a table/codebook with feature vectors and labels. We want to take as little training data as possible, but have it be sufficient to build a good codebook to predict location. The feature vectors are the RSS received from the U anchors, while the labels define the target's (x, y) locations. New RSS

values are obtained from the U anchors in the testing stage, and a location estimate has to be determined.

2.3.1. Training of RSS Fingerprinting Technique with Real Data

In developing the fingerprinting codebook, we separate an arbitrary area into R_m number of rooms. A reference grid is created. Each point in the grid is labeled according to room number and its (i, j)th position in the grid. The (i, j) fingerprint in room k is defined as,

$$\boldsymbol{F}_{i,j,k} = \begin{bmatrix} \nu_{i,j,1} & \nu_{i,j,2} & \cdots & \nu_{i,j,U} \end{bmatrix}^T \tag{11}$$

The origin of the coordinate system is the leftmost corner point of the geographical area. The position vector for the (i, j) reference point in room k is defined as:

$$D_{i,j,k} = [x_{i,j,k} \;\; y_{i,j,k}, \;\; z_{i,j,k}]^T \tag{12}$$

During the training stage, received signal strength is collected at all the (i, j) points in all k rooms. This data collection was carried out for a duration of five minutes at each of the reference points and is averaged in time. The data is stored in a codebook $\mathcal{F}$. The (i, j, k)th entry of the codebook can be accessed as $\boldsymbol{F}_{i,j,k} = \mathcal{F}[i, j, k]$. The signal strength received from the uth anchor at the (i, j) reference point in room k can be accessed through $\mathcal{F}[i, j, k][u]$. Note that:

$$\boldsymbol{F}_{i,j,k}[u] = \mathcal{F}[i, j, k][u]$$

Similarly, the position vectors are stored in a distance codebook defined as $\mathcal{D} = D_{i,j,k}$. The (i, j, k)th entry of the codebook can be accessed as $\boldsymbol{D}_{i,j,k} = \mathcal{D}[i, j, k]$.

2.3.2. Specifications of Area of Interest

We evaluated our k-NN-based localization technique in a home environment. The environment had an area of 10.6×7.4 m, which was divided into $R_m = 7$ rooms. Within this area, four beacons were placed, and 19 reference points were selected. RSS information is collected at each reference point for a duration of five minutes in order to form the signal strength fingerprint. The resulting data at each specific reference point is averaged and placed in a codebook. The figure in Appendix C gives coordinates of the beacon locations and it also provides the positions of the reference points used for building the fingerprinting codebook.

2.3.3. Real-World Validation

For validation, the target was placed at five different test locations and left there for approximately 45 s each. Note that the target is a Raspberry Pi, which is attached to the human body. At each location, RSS is collected at the target from the U anchors. The RSS received from each anchor is averaged down to 4 Hz. If no packets are received from the uth anchor, the RSS from that anchor is set to $\nu_u = \nu_{min}$, where ν_{min} is the minimum possible RSS. The RSS at the target during testing is defined as:

$$\boldsymbol{G} = \begin{bmatrix} \nu_1 & \nu_2 & \cdots & \nu_U \end{bmatrix}^T \tag{13}$$

The process of extracting position estimates is described by Algorithms 1 and 2. Algorithm 1 specifies the procedure to determine the closest reference points to the target. These reference points are obtained by comparing the received RSS signatures with the RSS signatures in the codebook. These comparisons are through the Euclidean norm. Note that different RSS signatures in the codebook are associated with different reference points. Furthermore, note that the number of reference points returned (W) is a parameter that can be optimized depending on the environment. In this work, $W = 3$ is used. Algorithm 2 returns the centroid of the closest reference points. In this algorithm, the position vector is initialized as a zero vector. Subsequently, the closest reference coordinates are sequentially

summed. This cumulative sum is divided by W. This centroid is the position estimate of the target.

Algorithm 1 Generate Closest Fingerprints.

$\forall i, \ \forall j, \ \forall k$ Compute $\left\| \boldsymbol{F}_{i,j,k} - \boldsymbol{G} \right\|$

Sort then store the indices of the W smallest values in the set $\mathcal{W}$.

return $\mathcal{W}$

Algorithm 2 Get Position Estimates Using K-Nearest Neighbors.

Require: $\hat{\boldsymbol{D}} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^T$

while $w < W$ **do**

Get $(i, j, k) = \mathcal{W}[w]$

$\tilde{\boldsymbol{D}} = \mathcal{D}[i, j, k]$

$\hat{\boldsymbol{D}} = \hat{\boldsymbol{D}} + \tilde{\boldsymbol{D}}$

end while

$\hat{\boldsymbol{D}} = \hat{\boldsymbol{D}} / W$

return $\hat{\boldsymbol{D}}$

3. Materials and Methods for Kinematics Estimation

In this section, we discuss our kinematics reconstruction algorithms and the methods for their experimental evaluation.

3.1. Overview

Briefly, we used our dataset and our motion inference algorithms [60] to generate the machine learning models for the upper body motion inference. The full-body kinematics contains information for 23 segments, while the upper body contains information for 15 segments. We aimed to predict the upper body kinematics using only information (orientation and acceleration data) from 3 segments, with the measured upper body kinematics of all 15 segments as the ground truth.

A summary of the pipeline for our work is presented in Figure 7. The top of the figure shows how we train our machine learning models. The Virginia Tech Natural Motion Dataset contains kinematic data for the whole body. We extracted just the upper body, and then used motion sequences of orientation and acceleration data from only three segments (pelvis and forearms) as inputs to the machine learning model. The model predicts the orientations of all 15 segments of the upper body, with the ground truth values from the dataset.

Following the creation of the machine learning models, we captured $N = 4$ participants' full-body kinematics using the XSens MVN Link suit. Simultaneously, we used three XSens DOT sensors (standalone IMUs) to capture orientations and accelerations from the pelvis and forearms. We used the newly captured dataset as the test set for our work: we inferred the predicted upper body kinematics based on (1) orientations from the XSens MVN system for the three sparse body segments (pelvis and forearms), and corresponding sensor accelerations, and (2) orientations and accelerations from the XSens DOT sensors. We compared the inferred upper body kinematics from each of these to the ground truth (15 segments of XSens MVN).

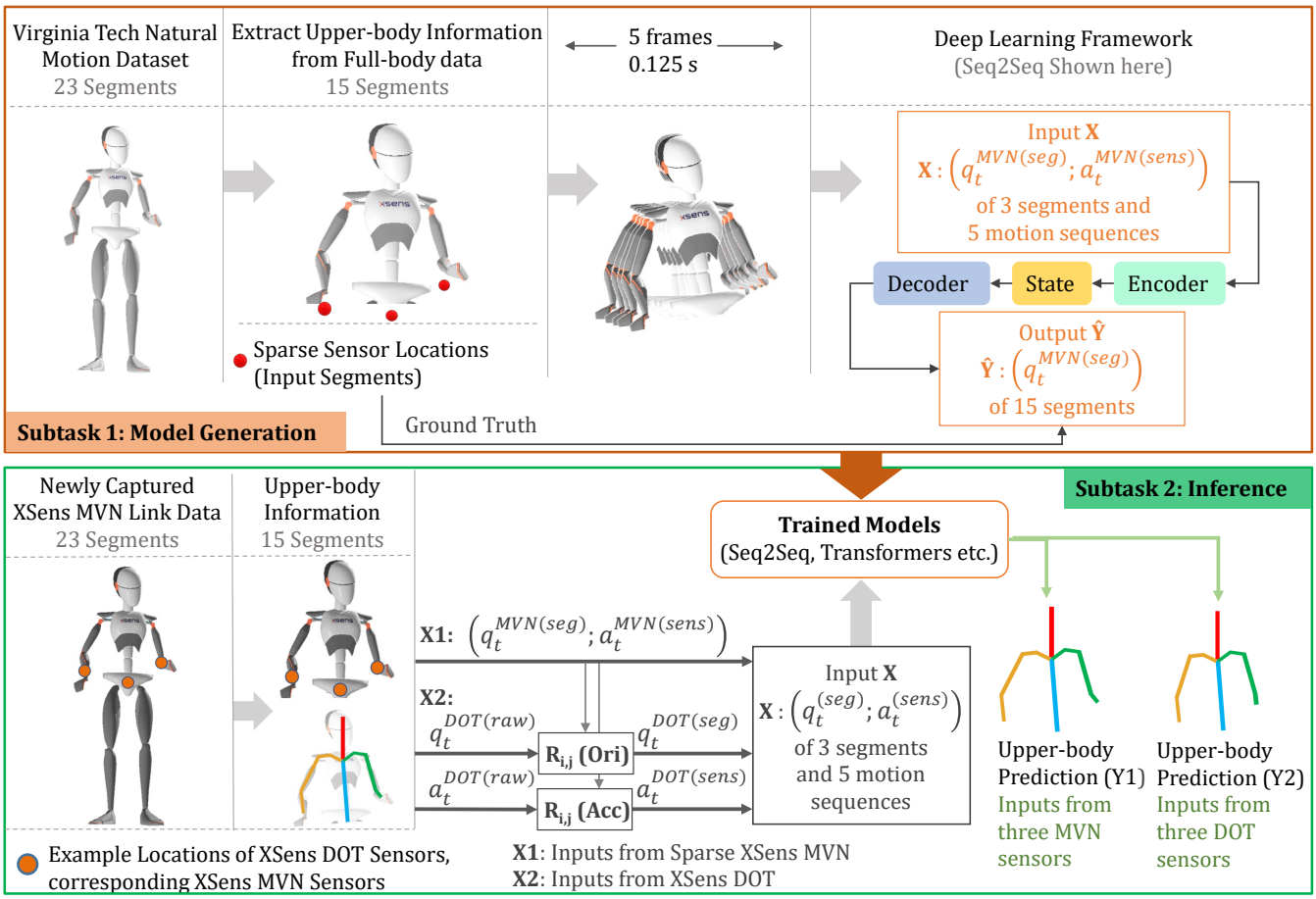


Figure 7. Subtask 1: We used the Virginia Tech Natural Motion Dataset to train our deep learning models. We used three sparse segments (pelvis, right forearm, left forearm), passed five frames of segment orientation ($q_t^{MVN(seg)}$) and sensor linear acceleration data ($a_t^{MVN(sens)}$) into a neural network, and then predicted upper body segment orientations for those five frames. Subtask 2: For Inference, we used newly captured XSens data (upper body; 15 segments) for the ground truth. We used two sets of sparse inputs: 3 XSens DOT sensors (**X1**), and 3 XSens MVN segments (**X2**). The machine learning models from Subtask 1 produced two sets of output upper body kinematics for the two sets of inputs. We then compared the predicted kinematics with the ground truth upper body information from the newly captured XSens MVN Link data. The data used for inference from the MVN was similar to that in Subtask 1; for the DOT sensors, the raw orientation ($q_t^{DOT(raw)}$) and acceleration ($a_t^{DOT(raw)}$) were calibrated to match the MVN coordinate system by multiplying with a rotation matrix $\mathbf{R_{i,j}}$.

3.2. Training Dataset Description

The Virginia Tech Natural Motion Dataset [70] is an enriched dataset of full-body human motion. The data was captured using an XSens MVN Link and includes more than 40 h of unscripted daily life motion in the open world.

The XSens MVN Link suit collects synchronized inertial sensor data from 17 IMU sensors placed in different segments of the body. The data collected from XSens (17 sensors) have reduced magnetic disturbance via a specialized Kalman filter design, and are post-processed to construct accurate human kinematics of 23 segments. The XSens MVN captures full-body kinematics within 5° of absolute mean error compared to an optical motion capture system for various tasks, including carrying, pushing, pulling, and complex manual handling [66,71–74]. The data includes measurements for segment position, segment linear velocity, both sensor and segment linear acceleration, both sensor and segment orientation, and segment angular velocity and acceleration.

The data were collected from 17 participants, where 13 participants were Virginia Tech students, and 4 were employees of a local home improvement store. Fourteen were male, and three were female. Participants were asked to perform many routine works and material handling tasks, including walking, carrying, pushing, pulling, lifting, and complex manipulation. While generating the deep learning models in this paper, we used orientation and acceleration data of the motion dataset participants {P1, P2, P3, P4, P5, P6, P8, P9, P13, W1, W2, W4} for training, {P10, P12, W3} for cross-validation, and {P7, P11} for testing. Here, 'P' refers to Virginia Tech participants, with ranges from P1–P13; 'W' refers to workers, with ranges from W1–W4. Details on data collection, data quality, and the role of each participant are documented in [60].

3.3. Subtask 1: Model Generation

3.3.1. Training Inputs and Outputs

In this paper, we studied upper-body motion inference, where only the kinematics of the upper body were predicted. We started the training subtask by extracting upper-body information from our training dataset. The XSens MVN Link generates a skeleton of 23 "segments" for the full body, where the first 15 segments are considered as upper body segments. In addition to providing the final body model of 23 segments, the XSens MVN Link also provides the raw data collected from each of the sensors placed on the body. In our previous work [60], we used the orientation and acceleration of sparse segments from the final reconstructed model. Here, we used the linear acceleration of one of the actual sensors from the XSens system ("sensor acceleration") in combination with the orientation of the reconstructed skeleton segments ("segment orientation") in our study.

The upper-body inference task was framed as a sequence-to-sequence problem. We entered a sequence of three segment orientations (pelvis, right forearm = RFA, left forearm = LFA) and the corresponding sensor accelerations to predict the orientation of all 15 segments of the upper body over the same sequence. To construct sequences, we down-sampled the upper-body orientation and acceleration data from 240 Hz to 40 Hz. We then took five frames of data as both input and output. Five frames of data at 40 Hz corresponds to a motion sequence that is 0.125 s long. Longer input and output sequences add computational complexity to the model without improved results, as discussed in [60]. We apply hyperparameter tuning to maximize neural network performance.

Possible rotational representations of the body segments include Euler angles, rotation matrices, exponential mapping, and quaternions [60]. Euler angle representation has some unavoidable issues namely locking and singularities [75]. Furthermore, rotation matrices incur some computational complexity [59]. An exponential map has been used in many prior works of human motion prediction [76–79]. However, for representing orientation, we used 4-dimensional quaternions for several well-defined reasons [60].

Before passing the parameters to the model, we normalize the segment orientation and sensor acceleration values of all segments with respect to the root (pelvis) segment. This normalization procedure is the same as in other works, such as [57,59,60]. The root (pelvis) segment orientation with respect to the global frame is $\boldsymbol{R}_{GP}$ ($\boldsymbol{R}$ refers to the orientation, G refers to the global reference frame, P refers to the pelvis segment reference frame). Then, normalized orientation of any segment with respect to the pelvis segment can be found using the following equation:

$$\boldsymbol{R}_{PB_i} = {\boldsymbol{R}^{-1}}_{GP} \cdot \boldsymbol{R}_{GB_i} \tag{14}$$

In Equation (14), B refers to body or segment frame, i refers to segment number (ranges from 1 to 15 for the upper body). Thus, $\boldsymbol{R}_{GB_i}$ is the ith segment orientation with respect to the global frame. The normalized orientation of segment i is $\boldsymbol{R}_{PB_i}$ (ith segment orientation with respect to the pelvis frame). Similarly, the sensor's normalized acceleration can be found using the following equation:

$$\bar{\boldsymbol{a}}_{BS_i} = {\boldsymbol{R}^{-1}}_{GP} \cdot \boldsymbol{R}_{GBi} \tag{15}$$

In Equation (15), $\bar{a}_{BS_i}$ refers to the normalized sensor acceleration for segment i. BS refers to the corresponding sensor frame of the segment frame B. After normalizing orientation and acceleration using Equations (14) and (15), we zero the mean and divide by the standard deviation of each feature in the training set. Since the validation and test data both simulate unseen data collected in the real-world, we made the assumption that they come from the same underlying distribution as the training data [60].

Briefly, for each task, the input to our model was 5 continuous poses of normalized segment orientation ('normOrientation') and normalized sensor acceleration ('normSensorAcceleration') for three segments (pelvis, RFA, LFA). The output of the model is the normOrientation value of 15 segments over the sequence of 5 poses.

3.3.2. Deep Learning Models

We used two deep learning architectures for human motion inference: sequence-to-sequence (Seq2Seq) and Transformers [60]. We used the same architectures for inferring upper-body motion from standalone XSens Dot sensors. We chose these architectures because human motion is naturally a temporal sequence, and Seq2Seq and Transformer architectures are efficient for predicting temporal sequences [78,80,81].

Sequence-to-sequence (Seq2Seq) has proven to be successful in neural machine translation [82] and other applications in natural language processing. Seq2Seq models consist of an encoder and a decoder. Furthermore, these models typically contain one or more layers of long short-term memory (LSTM) layers or gated recurrent unit (GRU) layers [62,83]. We also used a variant of the Seq2Seq architecture, where a bidirectional encoder was used [84,85]. Along with the bidirectional encoder we also used Bahdanau attention [85]. This attention mechanism helps to learn the important encoder hidden states.

Similar to the Seq2Seq architecture, a Transformer is also an encoder-decoder-based architecture. It can also be used for human motion inference [86] and other applications in natural language processing [87–91]. Unlike Seq2Seq models, it does not have recurrent layers. We made two models using the Transformer architecture: using a bidirectional encoder, which we refer to as 'Transformer Encoder'; and using both an encoder and decoder, which we refer to as 'Transformer Full'. More detail of the Transformer architecture and exact implementation can be found in the original paper [86] and two helpful tutorials [92,93].

In summary, we used two deep learning architectures (1) Seq2Seq (2) Transformers. From these architectures, we prepared four models (algorithms). We refer to these models as (1) Seq2Seq (2) Seq2Seq (BiRNN, Attn.) (3) Transformer Encoder, and (4) Transformer Full.

3.3.3. Training Parameters, Hyperparameter Tuning, and Performance Matrices

We generated the aforementioned models using PyTorch [94]. We conducted hyperparameter tuning using a training and cross-validation set. For each model, we used the same training/validation split. We placed P1, P2, P3, P4 P5, P6, P8, P9, P13, W1, W2, and W4 in the training set (Here, P = Virginia Tech participants and W = worker). In the validation set, we placed P10, P12, and W3. In total, we used 882,452 and 318,484 sequences for training and validation, respectively. We used a V100 GPU and AdamW optimizer with a learning rate of 0.001. Other details of the hyperparameters are provided in Table 2. We used mean absolute error (MAE) as the training loss function.

$$MAE = \frac{1}{mn}\sum_{j=1}^{m}\sum_{i=1}^{n}|\hat{q}_i - q_i| \tag{16}$$

In Equation (16), $\hat{q}_i$ is the predicted segment quaternion, q_i is the ground truth segment quaternion, n is the number of segments in the body being predicted (15 for the upper body), and m is the number of frames in the output sequence (5 frames).

Table 2. Important training parameters used in different deep learning models. In addition to these parameters, both the Transformer models used tuning parameters $\beta_1 = 0.95$ and $\beta_2 = 0.99$.

Parameters	Deep Learning Models			
	Seq2Seq	**Seq2Seq (BiRNN, Attn.)**	**Transformer Encoder**	**Transformer Full**
Batch Size	32	32	32	32
Sequence Length	30	30	30	30
Downsample	6	6	6	6
In-out-ratio	1	1	1	1
Stride	30	30	30	30
Hidden-Size	512	512	N/A	N/A
Number of Epochs	3	3	3	3
Dropout	0.1	0.1	0.1	0.1
Number of Heads	N/A	N/A	21	4
Number of Layers	N/A	N/A	2	4
Feedforward Size	N/A	N/A	200	2048

3.3.4. Training Performance Evaluation

For evaluating training performance, we used separate test sets (never used for training or cross validation). Our model evaluation test set came from participants P7 and P11. We used the mean angle difference $\bar{\theta}$ between the ground truth orientation and predicted orientation as a performance matrix of our models. We used the following equation to calculate $\bar{\theta}$ (in degrees).

$$\bar{\theta} = \frac{360}{\pi mn} \sum_{j=1}^{m} \sum_{i=1}^{n} |< \hat{q}_i, q_i >| \tag{17}$$

In Equation (17), q_i is the ground truth quaternion and $\hat{q}_i$ is the predicted quaternion for each segment, i is the index of the individual body segments, j is the index of the frames in the output, n is the number of segments (15 for the upper body), and $<\cdot, \cdot>$ is the inner product between two quaternions.

For visualization, we use a forward kinematics solver to plot a line model of the human upper body from the normalized orientation output. The forward kinematics solver uses the segment orientations and then multiplies by a single participant's segment lengths taken from an XSens MVNX file [60]. We used the following equation to perform forward kinematics given the orientation of the segment:

$$\mathbf{P}_{G_i}^{segment} = \mathbf{P}_{G_i}^{origin} + \mathbf{R}_{GBi} \cdot \mathbf{X}_i^{segment} \tag{18}$$

In Equation (18), $\mathbf{P}_{G_i}^{segment}$ is the position of the target segment's (ith segment) endpoint, $\mathbf{P}_{G_i}^{origin}$ is the position of the origin, and $\mathbf{X}_i^{segment}$ is the segment's length. As before, G refers to the global reference frame and B refers to the segment's reference frame.

Although normalization improves generalization, we multiplied by the orientation of the pelvis to view the posture as it would be viewed without normalization for qualitative evaluation. We used the following equation on the all predicted poses:

$$\boldsymbol{R}_{GB} = \boldsymbol{R}_{GP} \cdot \boldsymbol{R}_{PB} \tag{19}$$

3.4. Subtask 2: Inference

3.4.1. Test Dataset Overview

As discussed before, we wanted to compare the performance of sparse sensor configurations with three sensors derived from the XSens MVN system versus the performance of standalone sensors. Therefore, we collected data using XSens DOT sensors along with the full XSens MVN Link suit. We collected data from $N = 4$ new participants (2 males, 2 females; ages 23.0 ± 2.3 years). All subjects provided informed consent (Virginia Tech IRB #18-877). After putting on the required sensors, participants were asked to perform

some activities of daily living (ADL), listed in Table A1. The data collection was performed in a simulated house environment. Details of the data collection are discussed in the following section.

3.4.2. Data Collection

At the beginning of each experiment, the experiment rooms were prepared with the supplies required to perform the activities (full list in Table A1). Then, the participants put on the full XSens MVN Link system. After wearing the suit, four XSens DOT sensors ("DOT sensors") were secured on top of the Link sensors or the Link suit with tape (see Figure 8). Three DOT sensors were taped on top of the XSens MVN sensors on the pelvis, right forearm, and left forearm; these were sensors that corresponded to the sparse segments in our machine learning framework. The fourth DOT sensor was placed on the left side of the hip, which did not have a corresponding XSens MVN sensor. Complete details of the setup for data collection are presented in Figure 8.

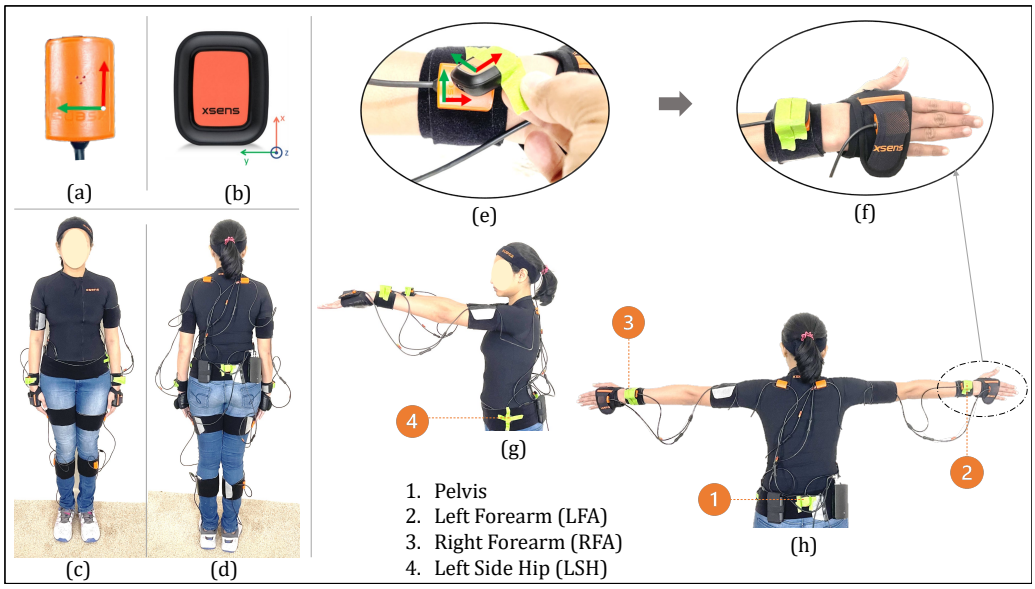


Figure 8. The data collection process using XSens MVN and XSens DOT sensors. In (**a**,**b**), we show the local coordinate system of the XSens MVN sensor and DOT sensors, respectively. In (**c**,**d**), we present front and back views of a participant wearing both sensor systems. In (**e**–**h**) we show detail of the locations of the XSens DOT sensors.

We recorded data with a rate of 240 Hz with XSens MVN and with a rate of 60 Hz with the DOT sensors. The DOT sensors were programmed to collect orientation (quaternions) and acceleration. Later, we downsampled data from both sensors to a rate of 40 Hz and synchronized them manually.

3.4.3. Study Design

Placing an IMU sensor at the back of the pelvis is quite popular in kinematic inference from sparse sensors (e.g., in [58,59,61]). However, we assumed for practical applications like stroke rehabilitation, that it might be uncomfortable for a patient to wear a sensor on their back for an extended period of time. To investigate solutions to this, we used two configurations to compare the accuracy of upper body inference. As presented in Figure 8, we placed four DOT sensors to formulate two configurations (Figure 9). For Configuration 1, we used DOT sensors at the pelvis, LFA, RFA segments. For Configuration 2, we use a sensor on the left side of the hip (LSH) instead of the pelvis sensor in Configuration 1.

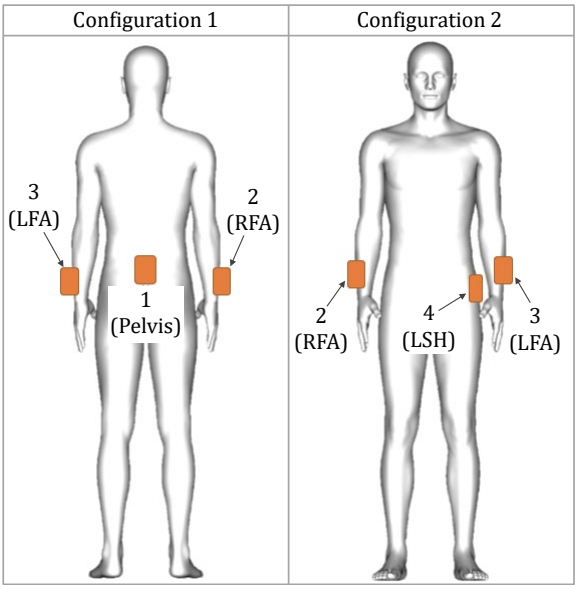


Figure 9. Diagram of the two different possible sensor configurations explored in this paper. Both of them have wrist-mounted sensors in the same locations. On the left, Configuration 1 has the pelvis sensor at the person's back, which is coincident with the XSens system pelvis sensor. On the right, Configuration 2 has the pelvis sensor at the person's left hip. LFA = left forearm, RFA = right forearm, SH = left side of hip.

In our study, as the ground truth, we used upper body (15 segments) orientation information from the full XSens MVN suit. We then performed motion inference using (a) three sparse segment configurations derived from the XSens MVN, (b) using three DOT sensors in Configuration 1, and (c) using three DOT sensors in Configuration 2.

3.4.4. Mathematical Framework: Inference Inputs and Outputs, and Sensor Calibration

With the two configurations of the DOT sensors in Figure 9, we mapped the orientation of the three DOT sensors to the three XSens MVN segments (since the segment orientations are inputs to our machine learning models). Similarly, we map the accelerations of the three DOT sensors to the corresponding XSens MVN sensor accelerations. We define two types of mapping functions to translate the DOT measurements to the MVN model, considering two cases: a variable mapping function that is customized for each trial, and a fixed mapping function that is the same across all participants.

For the variable mapping, we mapped the DOT sensor orientation and acceleration in two steps. In the first step, we mapped orientation (DOT sensor to MVN segment), and in the second step, we mapped acceleration (DOT sensor to MVN sensor). For orientation mapping, we assumed that a fixed rotation matrix (mapping function) existed between the DOT sensors and corresponding XSens MVN segment for each individual recording session. Similarly, for acceleration mapping, we assumed that a fixed rotation matrix existed between each DOT sensor and the corresponding XSens MVN sensor.

That means the mapping functions (orientation and acceleration) on a particular day (or recording session) may not be the same as the next day. We made this assumption because, for each recording session, the XSens MVN system performs a local calibration. This calibration might be different for a different recording session. We mapped the orientation and acceleration of DOT sensors to XSens MVN using the following equations:

$$\mathbf{R_{i,j}}(\mathbf{Ori}) = ({R_{i,j}}^{MVN(seg)})_n \cdot ({R_{i,j}}^{DOT(raw)})_n{}^{-1} \tag{20}$$

$$({R_{i,j}}^{DOT(seg)}) = \mathbf{R_{i,j}}(\mathbf{Ori}) \cdot ({R_{i,j}}^{DOT(raw)}) \tag{21}$$

$$\mathbf{R_{i,j}}(\mathbf{Acc}) = ({R_{i,j}}^{MVN(sens)})_n \cdot ({R_{i,j}}^{DOT(raw)})_n{}^{-1} \tag{22}$$

$$({a_{i,j}}^{DOT(sens)}) = \mathbf{R_{i,j}}(\mathbf{Acc}) \cdot ({a_{i,j}}^{DOT(raw)}) \quad (23)$$

In Equations (20)–(23), $({R_{i,j}}^{MVN(seg)})$ and $({R_{i,j}}^{MVN(sens)})$ are the MVN segment orientation and MVN sensor orientation of the ith segments (pelvis, LSH, LFA, RFA, etc.) from the jth recording session. These values are rotation matrices corresponding to the orientations (quaternions) of each segment. Similarly, $({R_{i,j}}^{DOT})$ is the orientation of a DOT sensor and $({a_{i,j}}^{DOT})$ is the linear acceleration from a DOT sensor. Values with (raw) superscripts are the raw DOT sensor data, while values with (seg) and $(sens)$ have been calibrated to match the MVN segment and sensor data, respectively. For both MVN and DOT data, values with n subscripts were those corresponding to a particular frame n that we used for calibration. $\mathbf{R_{i,j}}(\mathbf{Ori})$ and $\mathbf{R_{i,j}}(\mathbf{Acc})$ are the desired orientation and acceleration calibration mapping functions (rotation matrices), respectively, for the ith segment and jth recording session.

Synchronization of the DOT sensor and XSens MVN is crucial to determine the mapping functions $\mathbf{R_{i,j}}(\mathbf{Ori})$ and $\mathbf{R_{i,j}}(\mathbf{Acc})$. For synchronization, we first downsampled the DOT sensors and corresponding XSens MVN segments to a frequency of 40 Hz. We then carefully synchronized both sensor data with a standard starting and ending frame based on a sudden bump, which is visible in the acceleration data. To then find the orientation mapping function $\mathbf{R_{i,j}}(\mathbf{Ori})$, we picked a random single frame (n), took the value of $({R_{i,j}}^{MVN(seg)})_n$, and multiplied (matrix product) it with the inverse of the corresponding DOT sensor orientation $({R_{i,j}}^{DOT(raw)})_n{}^{-1}$. This is shown in Equation (20). Similarly, to find the acceleration mapping function $\mathbf{R_{i,j}}(\mathbf{Acc})$, we used the same frame n, took the value of $({R_{i,j}}^{MVN(sens)})_n$, and multiplied (matrix product) it by $({R_{i,j}}^{DOT(raw)})_n{}^{-1}$. This is in Equation (22).

Once we constructed the fixed mapping functions, we then used these mapping functions for all of the data collected in that session (Equations (21) and (23)), to map the orientation and acceleration of all frames of the XSens DOT sensor to XSens MVN coordinate system. Finally, we used the mapped data $({R_{i,j}}^{DOT(seg)}, {a_{i,j}}^{DOT(sens)})$ as the input to the models. These relationships can also be seen in Figure 7.

For the fixed mapping, we assumed that a fixed rotation matrix (mapping function) exists between the DOT sensors and the corresponding XSens MVN segment and sensors, irrespective of the recording session. In other words, we assumed there exists a constant universal mapping function between DOT sensors and XSens MVN (sensors and segments). We made this assumption to investigate a generalized approach to using standalone IMUs for human motion inference. We found this fixed mapping function by averaging the variable mapping functions, using Equation (24).

$$\overline{\mathbf{R_{i,j}}} = quaternion\_average(\mathbf{R_{i,j_1}}, \mathbf{R_{i,j_2}}, \mathbf{R_{i,j_3}}, \dots) \quad (24)$$

In Equation (24), we simply average the mapping functions of different recording sessions using the quaternion averaging method [95]. We then used $\overline{\mathbf{R_{i,j}}}$ (refers to both the orientation and acceleration mapping functions) to map all DOT sensor data to XSens MVN data. In our study, we averaged the variable mapping functions from $j = 4$ recording sessions to estimate the fixed mapping function. Figure 10 shows the individual rotation matrices $({R_{i,j}}^{MVN(sens)})$ for each of the $j = 4$ recording sessions and the average rotation matrix from these four quaternions.

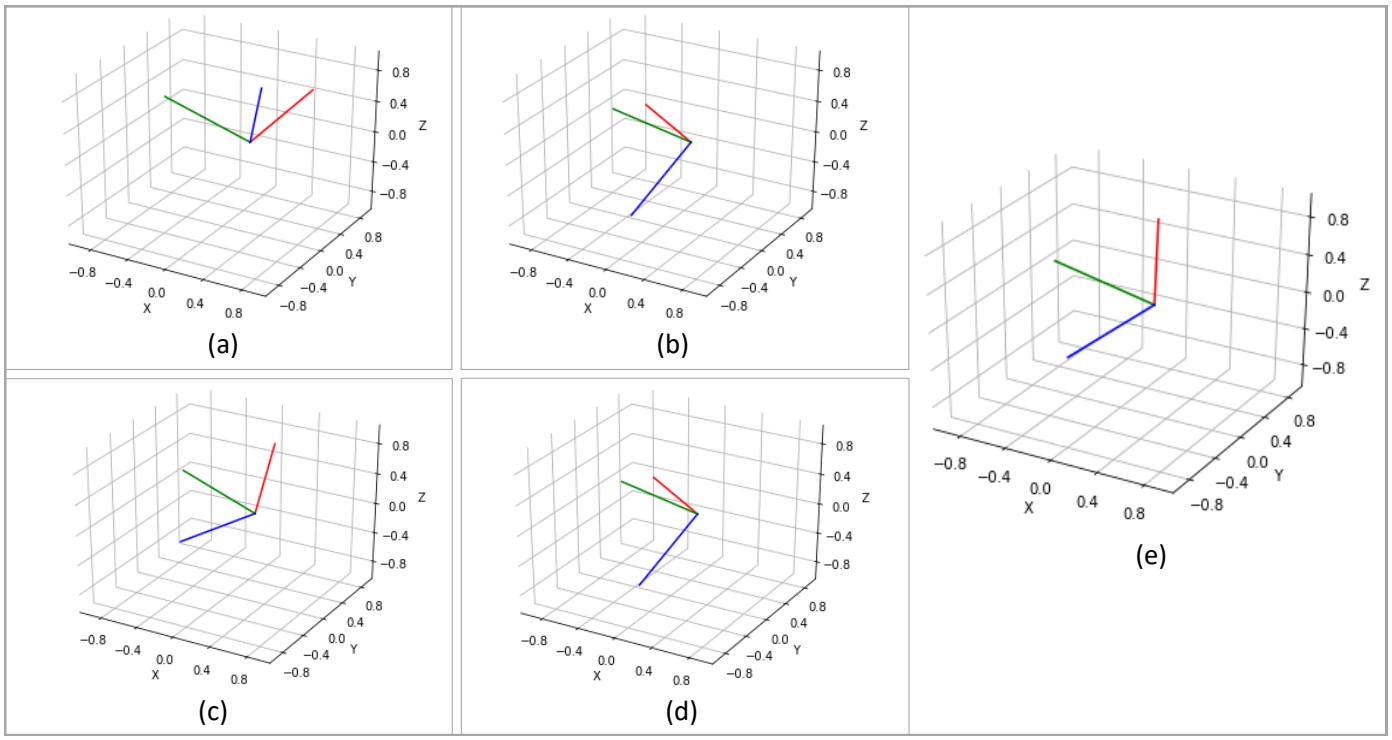


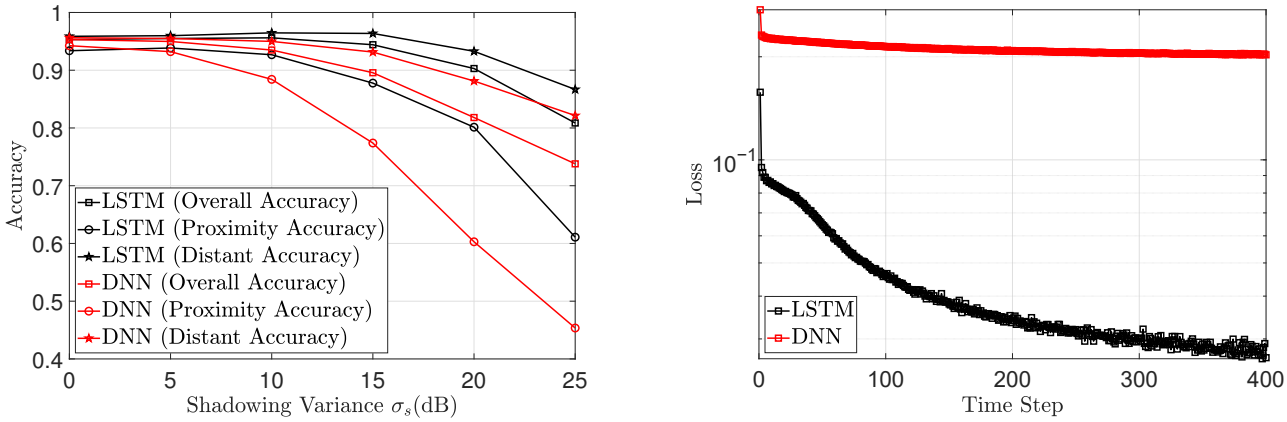
Figure 10. In (**a**–**d**), we plot the rotation matrix (variable mapping function) of XSens DOT sensors to MVN sensors for different persons on different recording sessions (only right forearm sensor is shown here). In (**e**), we plot the fixed rotation function (fixed mapping function). The fixed mapping function (using Equation (24)) is the quaternion average of the other four rotations. In each graph, the red, green, and blue lines correspond to the x, y, and z axes of each rotation matrix.

Inference using the sparse configuration of XSens MVN was straightforward. We used the segment orientation and sensor acceleration information of three sparse segments from the newly collected data to predict the upper body using the four machine learning models. However, for inference with standalone DOT sensors, we considered all possible combinations of the factors: deep learning models could be {Seq2Seq, Seq2Seq (BiRNN, Attn), Transformer Enc., Transformer Full}; the DOT sensors could be in {Configuration 1, Configuration 2}; and the Mapping Function could be {Variable Mapping, Fixed Mapping}.

4. Results

4.1. Localization Results: Proximity Reporting in Simulation

As simulation validation, we show the ability of the trained recurrent neural network (a DNN with a single LSTM layer) to withstand highly variable data; we also present the performance of a plain DNN. The anchor characteristics are presented in Appendix B. The path loss $PL_u(d_0)$ and path loss exponent ξ_u in that table describes the characteristics unique to a specific anchor. In Figure 11a, as the shadowing variance increased, σ_s, the accuracy of the LSTM degraded more slowly than with a simple DNN. More specifically, at a variance of $\sigma_s = 15$ dB, the proximity accuracy of the DNN was 78%, while the proximity accuracy of the LSTM was 87%. At the same variance, the distance accuracy of the DNN and LSTM was 89% and 95%, respectively. The training loss presented in Figure 11b indicates that the LSTM might have better performance since it converges to a smaller loss value.

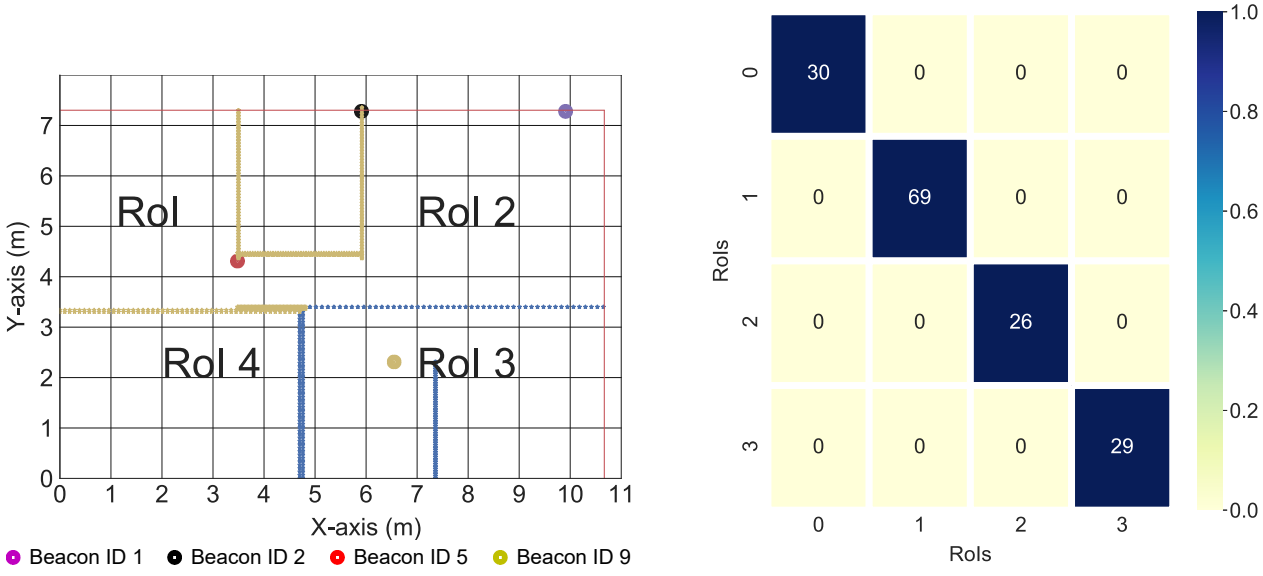


(**a**) (**b**)

Figure 11. (**a**) Accuracy versus shadowing variance. (**b**) Training loss versus time step.

4.2. Localization Results: Proximity Reporting with Real World Data

In this section, we validate the model-free neural network proximity reporting system with real data. The considered environment is an area of 10.6 × 7.4 m. The reference points and the beacons are placed as described in Section 2.3.2. The beacons and reference coordinates are also shown in Appendix C. However, unlike in Section 2.3.2, the area was divided into four regions of interest, as shown in Figure 12a. The red line indicates the wall separating the indoors and the outdoors. The yellow lines indicate walls separating various indoor regions, while the blue lines indicate a separation from one indoor region to the next without a wall. A three layered LSTM was trained to recognize each RoI. There were four inputs to the LSTM, each representing the mean RSSI values measured over 0.5 s from each of the four beacons. There were also four LSTM outputs, each representing the four RoIs. From Figure 12b, the LSTM was able to perfectly determine when the target was in each RoI.

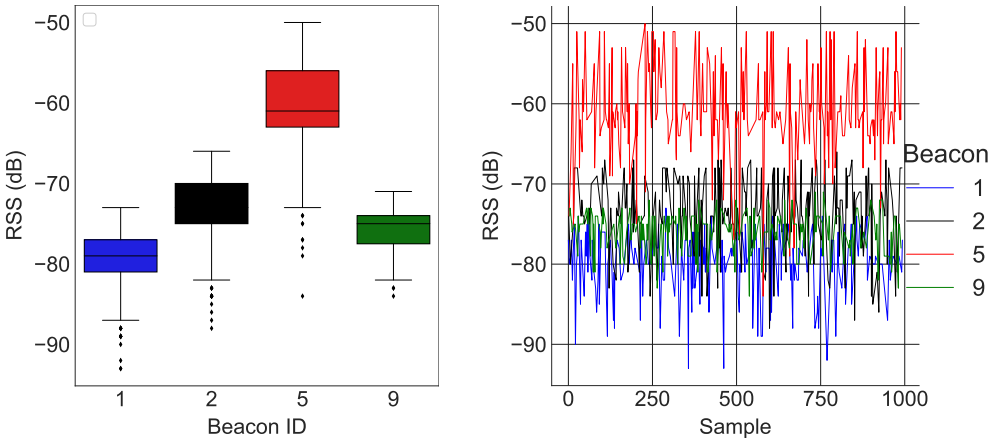


(**a**) (**b**)

Figure 12. (**a**) Floor plan, showing the beacon locations and the regions of interest; (**b**) confusion matrix showing accuracy of neural network-based proximity reporting.

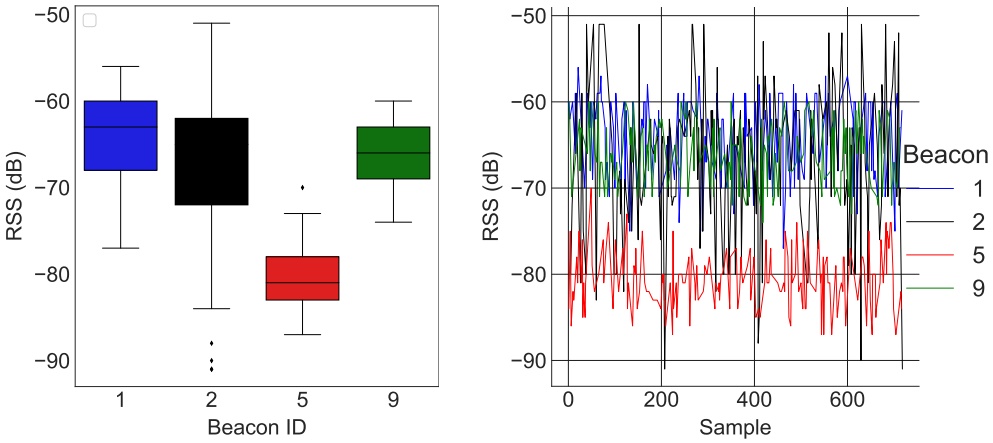
4.3. Localization Results: Positioning with Real World Data

In this section, we present real-world results using the *k*-NN algorithm presented in [29]. Two examples of RSS data at different locations are shown in Figures 13 and 14. The average values of the RSS were used to form the codebook for localization.



(**a**) (**b**)

Figure 13. RSS at Reference point 1. (**a**) Distributions of RSS for different beacons; (**b**) RSS versus sample number (samples are taken at 4 Hz).

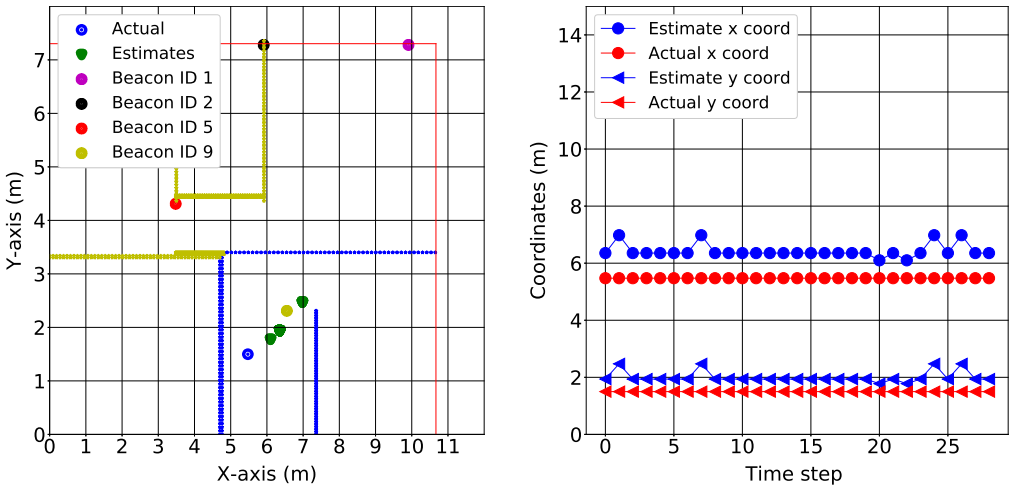


(**a**) (**b**)

Figure 14. RSS at Reference point 11. (**a**) Distributions of RSS for different beacons; (**b**) RSS versus sample number (samples are taken at 4 Hz).

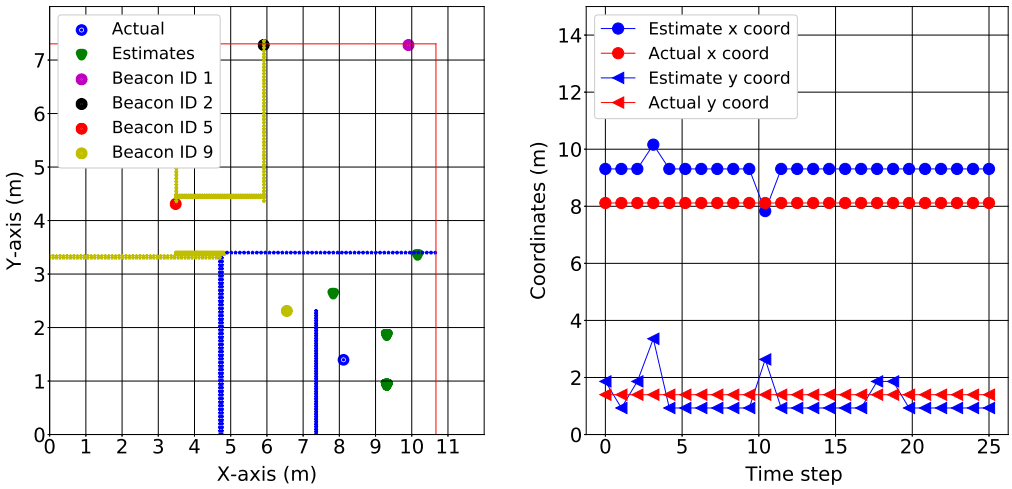
We provide test results for data collected in three of the seven rooms, as shown in Figures 15–17. The colored lines in these figures represent the demarcations separating the RoIs. These demarcations also affect the RSS in the form of shadowing. While the minimum mean square error of the position estimates was 1.78 m, the figures show that the target can be localized to an RoI. More specifically, the fingerprinting technique with real world data can predict what region of the house the target is located. This is crucial for smart health applications where knowing the section of the house that different motions are performed in gives cues about the purpose of those motions and which activities of daily living might need additional rehabilitation.

Figures 15b, 16b and 17b show the variation of the estimates over time in both the x coordinates and y coordinates. It is important to note the relationship between the spatial and the temporal view. For the kitchen, the algorithm produced varying estimates while the target was positioned at a fixed coordinate (see Figure 15a); this variation was captured best in the temporal view. In the temporal view (see Figure 15b), the estimates varied over the time step. This trend was also observed in the dining room (see Figure 16a,b). However, in the final test location (bedroom 2), the algorithm produced a stable estimate (see Figure 17a). This stability was validated by the temporal view in Figure 17b.



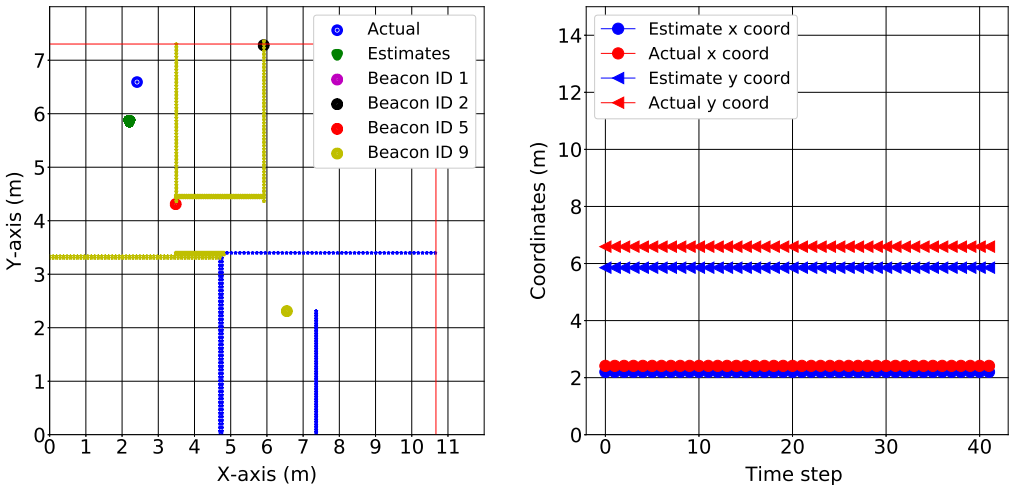
(**a**) (**b**)

Figure 15. Testing in the kitchen. (**a**) Floor plan view, and (**b**) location estimates over time.



(**a**) (**b**)

Figure 16. Testing in the dining room. (**a**) Floor plan view, and (**b**) location estimates over time.



(**a**) (**b**)

Figure 17. Testing in a bedroom. (**a**) Floor plan view, and (**b**) location estimates over time.

4.4. Motion Inference Results

Here, we first provide results on the performance of our algorithms. We describe the quantitative results for our algorithms, then we show the visualization of a few postures predicted by the models.

4.4.1. Quantitative Analysis

With the quantitative results, we first present the inference performance of our models using the VT Natural Motion Dataset. We then describe how the trained models performed with the new dataset using sparse segments of XSens MVN. Finally, we present the performance of our models using DOT sensors, considering the two configurations and the two mapping functions.

Test Performance Evaluation Using Sparse Segments of XSens MVN

We first present results using our new test set, as described in Section 3.4.1. Here, we expect similar results to our prior work [60], since we used the sparse data from XSens MVN. In Figure 18, we plot the angular error distribution combining all predicted segments for all four models, including the mean angular error for all models.

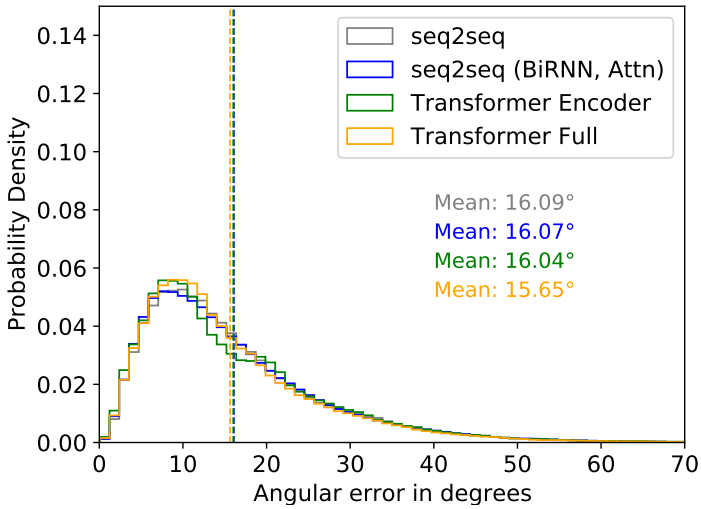


Figure 18. Distribution of the mean angular error for motion inference using the sparse segments of XSens MVN (from the new dataset).

Test Performance Evaluation Using XSens DOT Sensors

Next, we present inference results using the XSens DOT sensors considering different factors. Figure 19 shows the distribution of the mean angular error of the predicted segments relative to the ground truth segments for the two configurations we described in Figure 9. We consider the variable mapping function for the results shown in Figure 19. Overall, all the models performed similarly in both Configurations 1 and 2. However, results were slightly better in Configuration 1. Therefore, comparing the results of all configurations with the variable mapping, we used Configuration 1 and the Transformer Full model for further analysis, as these had the best results.

In Table 3, we present the results from the DOT sensors using the fixed mapping function. Here, average results were much better in Configuration 1 than Configuration 2. In Configurations 1 and 2, the transformer models had the minimum mean angular error, with values of $\sim 33^\circ$ in Configuration 1 and $\sim 43^\circ$ in Configuration 2.

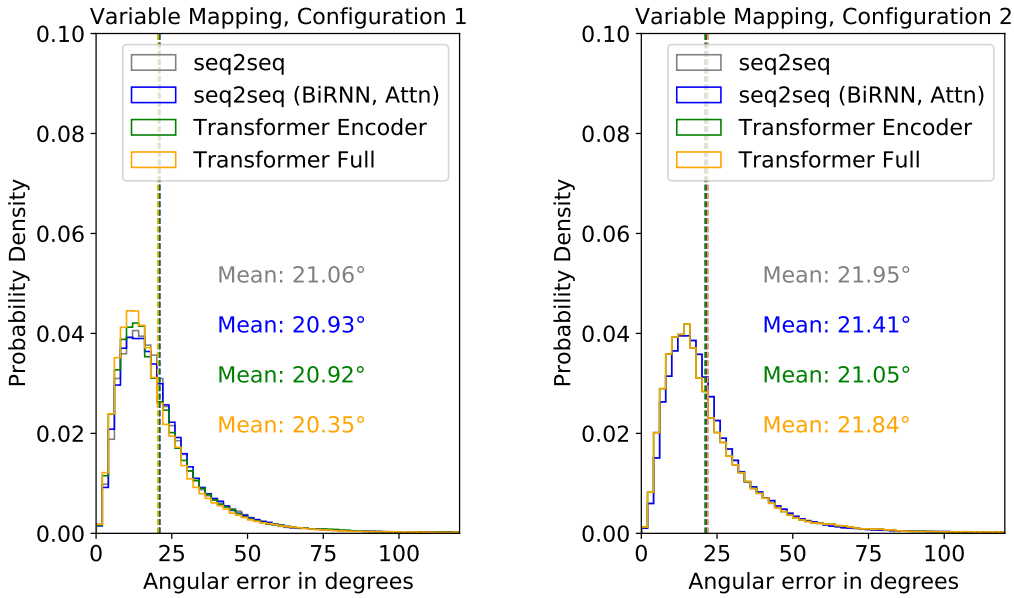


Figure 19. Angular error distribution of motion inference using Xsens DOT sensor with varying configurations (from the new dataset). Configuration 1 had the pelvis sensor on the back of the pelvis, next to the XSens sensor; Configuration 2 had the pelvis sensor on the side of the pelvis.

Table 3. Mean angular error of different models with varying configurations for inference using DOT sensors with the fixed mapping function (with the new test set).

Deep Learning Model	Fixed Mapping, Configuration 1	Fixed Mapping, Configuration 2
Seq2Seq	34.75°	44.85°
Seq2Seq (BiRNN, Attn)	33.07°	46.37°
Transformer Enc.	33.81°	42.92°
Transformer Full	32.65°	43.03°

Comparison of Segment-Wise Mean Angular Error of Predictions by XSens MVN and XSens DOT Sensors

In Figure 20, we compared inference results of the DOT sensors with results from sparse XSens MVN segments. We only compared the performance of the Transformer Full model. For DOT sensors, we present results with the variable mapping function and Configuration 1. There are 15 sub-figures for the 15 upper-body segments. For both XSens MVN and XSens DOT, we plot the mean angular error distribution for each segment relative to the ground truth. While the overall minimum mean angular error of prediction using XSens MVN and XSens DOT are $\sim$15.65° and $\sim$20.35°, respectively, for the Transformer Full model (see Figures 18 and 19), Figure 20 shows how these errors are distributed among the segments.

In most cases, the two inputs gave similar results, but using sparse segments of the XSens MVN performed several degrees better. Both inputs had relatively low mean angular errors for the first six segments (Pelvis, L5, L3, T12, T8, Neck), and the MVN inputs had low errors for the right and left forearms. For the XSens MVN, the maximum mean error occurred for the 'Head' segment, $\sim$29°. Noticeably, the XSens DOT had much higher mean errors for inferring motions of the 'Left Forearm' and 'Right Forearm', with errors of $\sim$24° and $\sim$26°, respectively; for comparison, the MVN inputs had errors of $\sim$3°.

We next computed histograms of the distribution of the joint angles measured in the test set (Figure 21). Specifically, we plot the left and right elbows and left and right shoulders. To find the joint angles, we took the angle between the two quaternions for the segments on either side of the joint. Therefore, the elbow angles were computed via the angle between the upper arm and forearm, while the shoulder angles were computed by the angle between the T8 segment (near the upper chest) and the upper arm. Note that this

method finds the smallest angle between the two quaternion orientations, so we do not distinguish between the three different degrees of freedom at the shoulder. Notably, both the XSens MVN and DOT inputs gave joint angle distributions very close to the ground truth for all of the angles investigated.

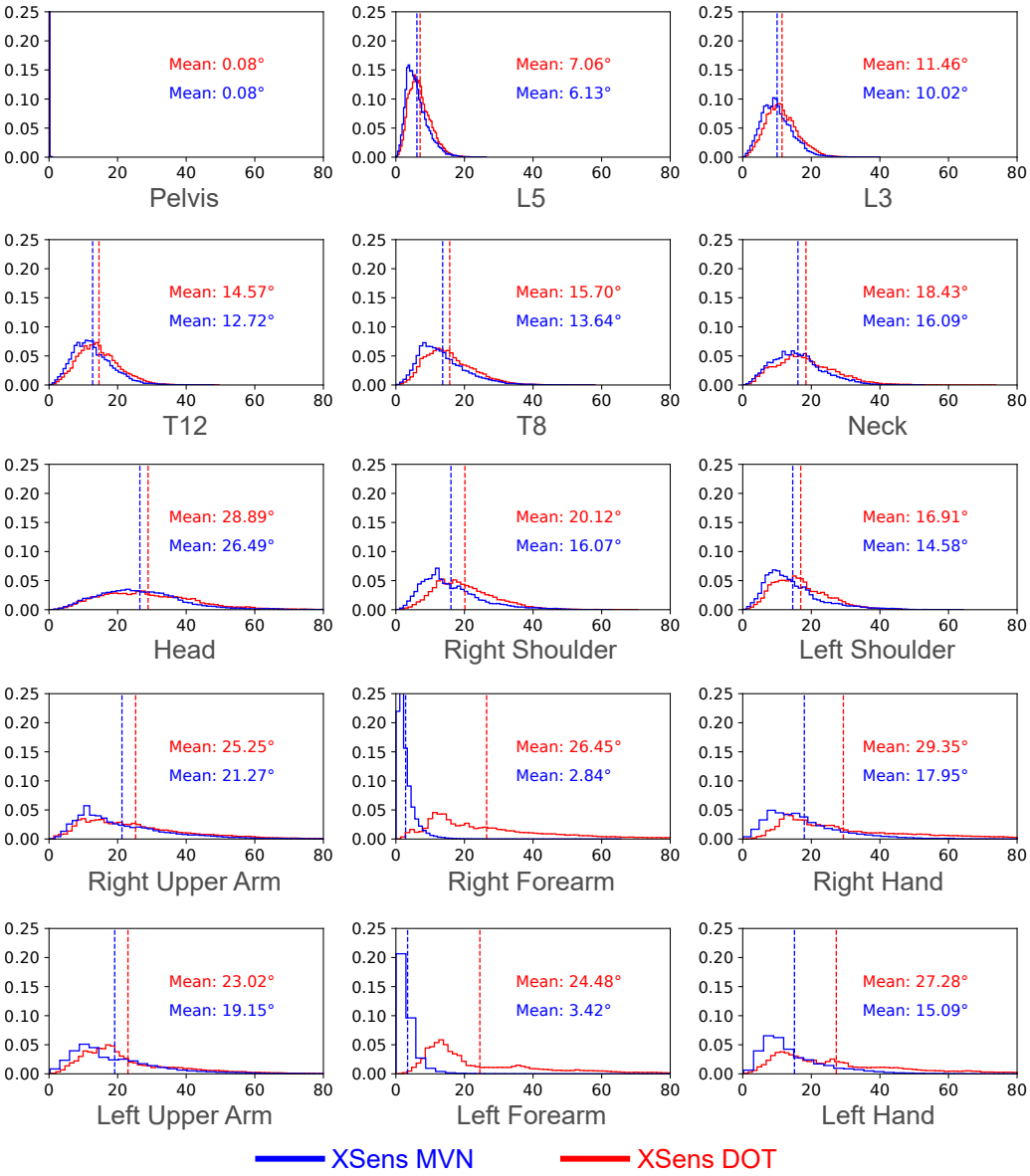


Figure 20. Segment orientation errors for each segment in the XSens model. *X*-axis is angle in degrees. Blue histograms show the angular error distribution for the MVN segments, while red histograms show the angular error distribution using the DOT sensors. The mean error is shown in each case.

Next, we computed histograms of the error between the ground truth joint angle and the joint angles predicted by either the MVN sparse segments or the DOT sensors (Figure 22). To find these values, we took the ground truth joint angles (as computed above) and subtracted from them the inferred joint angles (from the MVN and DOT sensors separately). In these graphs, negative values indicate that the inferred angle predicted a more acute angle than the ground truth. For the shoulder, negative values indicated that the arm was closer to the side than the ground truth. In each case, the mean joint angle error was less than 4.0° for both the MVN sparse sensors and the DOT sensors. The error distributions were approximately symmetric around zero in both cases.

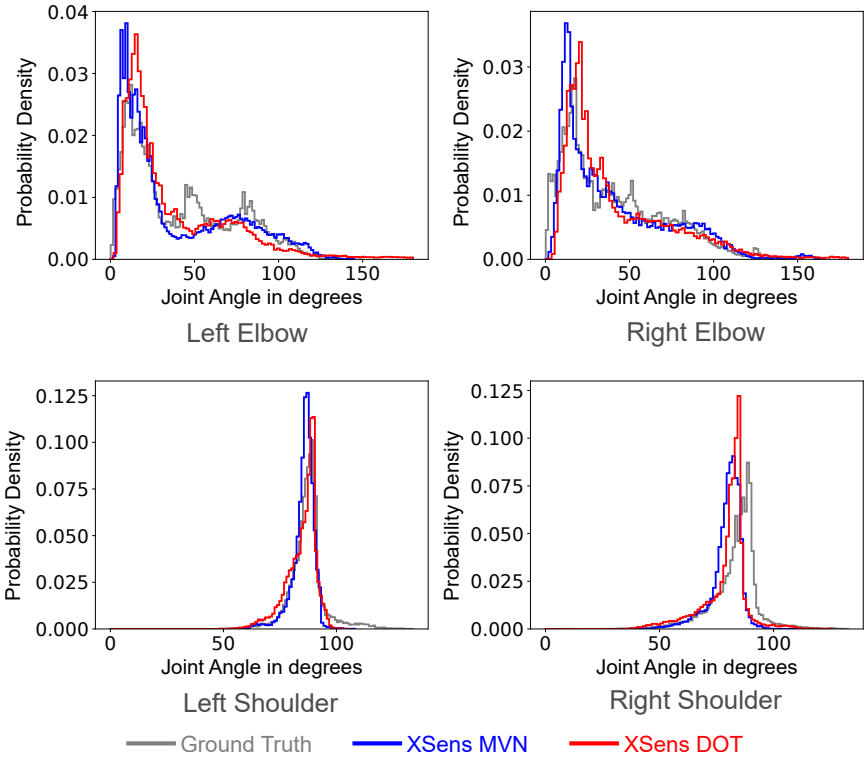


Figure 21. Joint angle distribution for selected joints. The blue histogram shows the joint angle distribution for the MVN segments, while the red histogram shows the joint angle distribution using the DOT sensors (variable mapping, Configuration 1). The grey histogram presents the ground truth for the corresponding joint angles. For the elbow, 0° corresponds to the arm straight.

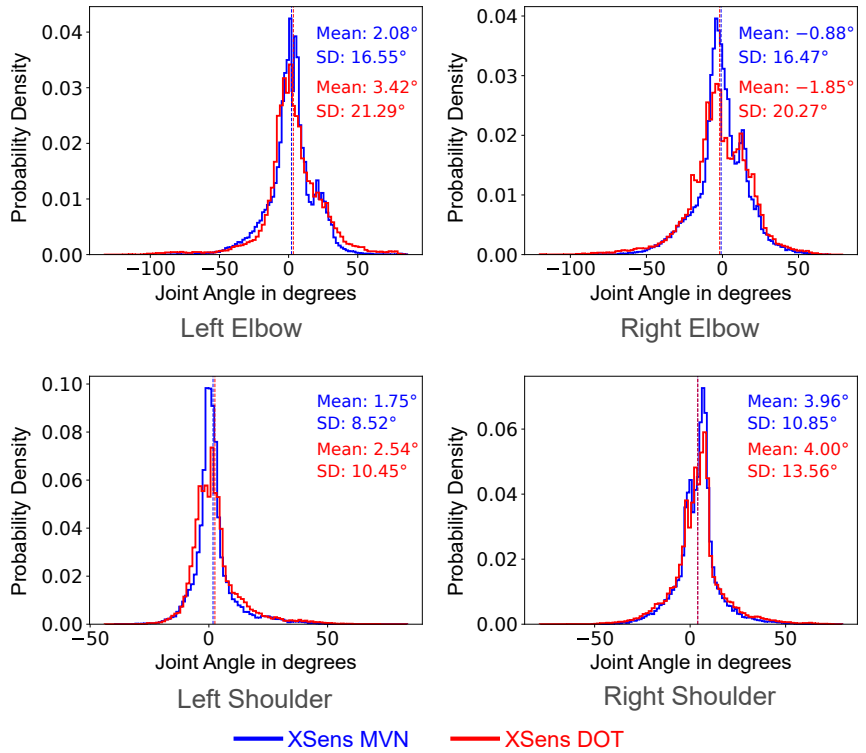


Figure 22. Joint angle error distribution for selected joints. The blue histogram shows the error distribution for the MVN segments, while the red histogram shows the error distribution using the DOT sensors (variable mapping, Configuration 1). The mean error and standard deviation are shown for each distribution.

4.4.2. Qualitative Analysis

Qualitative evaluation is performed in most of the studies of human motion inference [48,59,96] to give intuition into how well the reconstruction performs. Quantitative measures help analyze different aspects of the models' performance, and a visual evaluation is necessary to build intuition for how the models make predictions. We only evaluated a few poses to demonstrate our work.

Figure 23 presents four sample poses and the ground truth reconstructed using the XSens MVN Link system. We note that the actual human poses in Figure 23 correspond to slightly different times than the stick figures. The poses are representative of the activities listed in Table A1. The first pose shows vacuum cleaning, and the second pose shows folding laundry. The third pose is from organizing groceries, which is similar to picking something up from the ground. The fourth pose illustrates placing an object (either grocery/laundry) on a higher-level shelf.

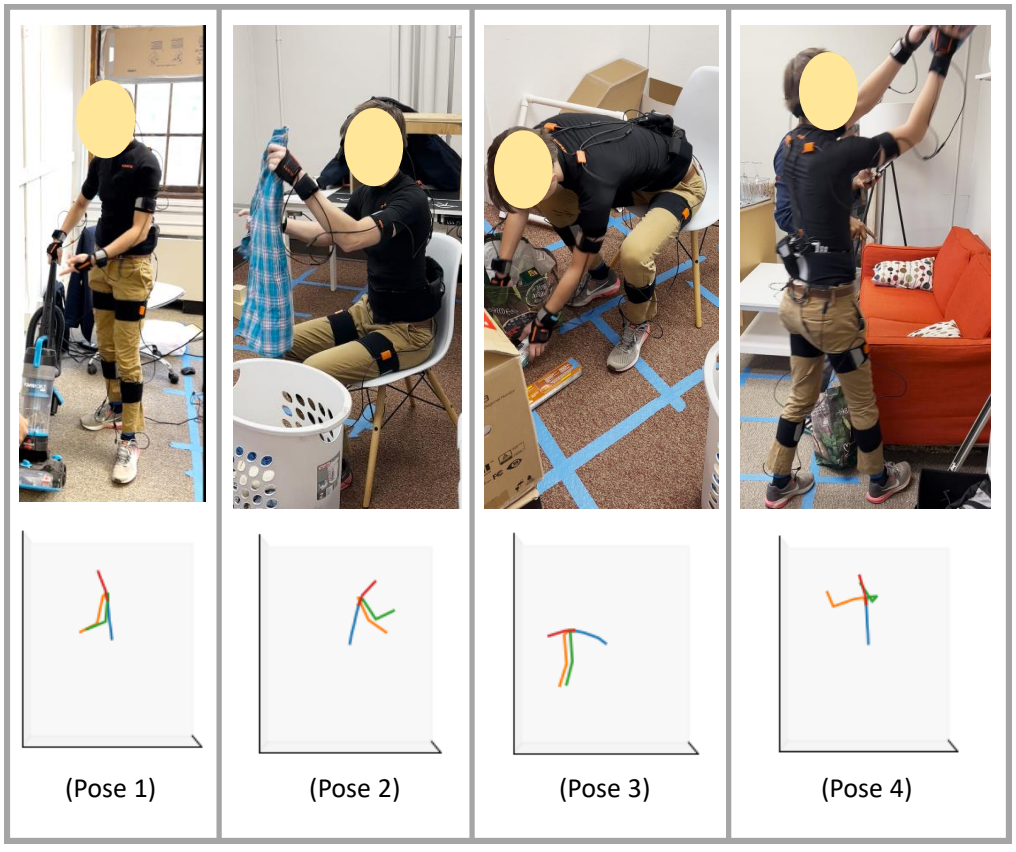


Figure 23. Figures in the first row are the samples of actual human poses during the different ADL tasks listed in Table A1. These include: vacuum cleaning (pose 1), folding laundry (pose 2), picking grocery items (pose 3), and putting objects on a high shelf (pose 4). In the second row, we present the skeleton model of the ground truth for the upper body for similar poses. The ground truth is reconstructed using the XSens MVN Link system. The actual human poses in the first row look slightly different than the ground truth poses in the second row because the photos correspond to slightly different times than the ground truth poses.

In Figures 24 and 25, we compare motion inference results using sparse segments of XSens MVN and XSens DOT (Configuration 1, variable mapping function). In each of the two figures, in the left-most column, we present the ground truth pose (as described in Figure 23), and on the right, we present inference results for both XSens MVN and XSens DOT from the four different machine learning models.

In the first pose (Figure 24 top), the person is standing and performing vacuum cleaning. Almost all the models performed well for both the XSens MVN and DOT sensors,

giving reasonable-looking results. Therefore, we expect good inference results for similar tasks where the person will be standing and doing other activities of daily living such as washing dishes in the kitchen, making food, or cleaning. In the second pose (Figure 24 bottom), the person folds laundry while sitting in a chair. This pose is similar to sitting for a meal or working on a study table or similar environment where the person does not need to bend much. Both sensor types again gave reasonable results. In the case of DOT sensors, the left elbow was inferred to be slightly more open than in the ground truth. Both sensor types show the right upper arm to be rotated slightly relative to the ground truth.

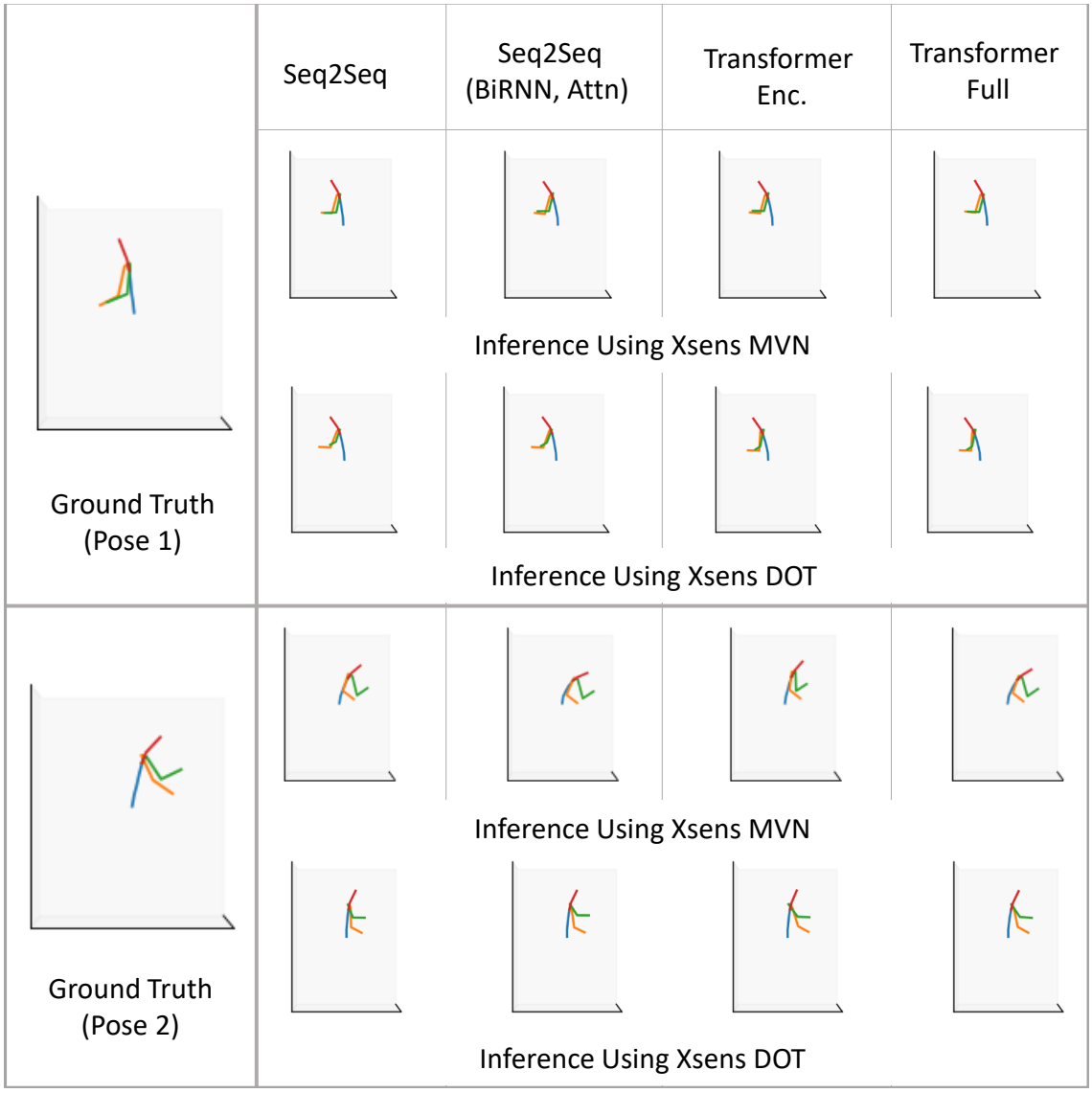


Figure 24. Qualitative evaluation for general human poses while standing (Pose 1) and sitting (Pose 2).

The third and fourth poses (Figure 25) were more challenging than the first two poses. In the third pose, the person bends more than 90°. This is similar to tasks such as picking up objects from the floor or organizing low objects. All models performed similarly for both the XSens MVN and DOT sensors, and gave reasonable outputs. In all of the models, the arms are not as far forward as in the ground truth. In the fourth pose, the person was reaching upward. Pose 4 was similar to organizing objects on a shelf, grabbing grocery objects from a refrigerator, placing laundry items in the closet, or similar tasks. The transformer models did not perform as well for the DOT sensors, but overall

all of the models performed reasonably. The MVN inference was slightly better than the DOT inference. Overall, the qualitative results resemble the quantitative evaluation of our models.

Figure 25. Qualitative evaluation is presented for challenging human poses while standing (Pose 4) and sitting (Pose 3). These poses are relatively challenging for inference since the person uses many hand or body movements.

5. Discussion

5.1. Discussion on Localization

5.1.1. Proximity Reporting

This work has developed model-free techniques for proximity reporting. The neural network takes as input RSS signatures from the beacons. During training, the neural network jointly learns the correlation among the beacons and the correlation between the target's position and received RSS. This learning approach circumvents the need to derive RSS thresholds for each beacon. The learning approach is validated in terms of its ability to detect whether the simulated patient is within a predefined range from the anchor, its ability to detect when the simulated patient is not within a predefined range from the anchor, and its ability to either place the simulated patient within the range from the anchor or to determine the absence of the simulated patient within a certain range from the anchor.

From the results presented in Figure 11a, all evaluated metrics deteriorate as the shadowing variance increases. This is intuitive, as the shadowing models the power fluctuation due to objects obstructing the propagation path between transmitter and receiver. A measurement from an obstructed beacon will most times have a reduced RSS value, giving the illusion that the target is much farther away than it actually is. This bias hampers any effect to accurately position the target. Figure 11 also depicts the advantage of accounting for the correlation between past and current measurements. The LSTM has better accuracy metrics than the DNN because it considers past measurements as well as future measurements when returning a proximity report.

5.1.2. Positioning with Real World Data

Figures 13a and 14a describe the time variation of the RSS from all of the anchors at the 1st reference point and 11th reference point, respectively. Clearly, the closest anchor has the highest mean RSS values. At the 1st reference point, the anchor with beacon ID 5 was the closest and had the highest mean RSS value (-69 dB). At the 11th reference point, the anchor with beacon ID 1 was the closest and had the highest mean RSS value (-63 dB). Figures 15a, 16a and 17a depict a few results showing the estimates and true positions. The estimates are roughly within the bounds of a room, which is likely sufficient for the interpretation of upper body kinematics. In [36], a multi-layer perceptron was used to achieve an accuracy of 2.82 m. In [97], a discriminant-adaptive neural network was developed for indoor positioning. A 23×30 m area was considered and a position accuracy of 2 m was achieved 60% of the time. In [28], a weighted k-NN approach was used for indoor positioning. For a similar 23×30 m area, an accuracy of 2 m was achieved 40% of the time. Considering all these works, our results also provide similar positioning accuracy. The positioning accuracy of our system varied from 1.3 m to 2.3 m.

5.2. Discussion on Motion Inference

Our results are well comparable to other previous work such as [57,59,61]. In [57], the authors used five sparse XSens MVN segments for predicting full-body poses, and compared six different configurations. Among them, configuration B was similar to our work. In configuration B, they placed sensors in the 'pelvis', 'left forearm', 'right forearm', 'left lower leg', and 'right lower leg' segments. The upper body in this configuration was comparable with our study (variable mapping with DOT sensor and sparse segment configuration of XSens MVN). Their estimates had an average joint angle error of $\sim 7^\circ$ and joint position error of ~ 8 cm for the full body, which is impressive. However, considering only the joints in the upper body, the mean joint angle errors were ~ 12–15° (Figure 5 in reference [57]), using five sparse sensors. In [59], the authors predicted skinned multi-person linear model (SMPL, [98]) parameters of a single frame using 20 past frames and five future frames at test time with a bidirectional LSTM network. They performed both online and offline evaluations. From Table 3 in reference [59], for offline evaluation after fine-tuning, their model estimated mean ($\pm$standard deviation) joint angle errors of $\sim 16^\circ \pm 13^\circ$ for the TotalCapture test dataset and $\sim 18^\circ \pm 12^\circ$ for the DIP-IMU dataset. In the recent work in [61], authors also used the SMPL parameters, and they performed both localization and motion inference using six standalone IMU sensors. Looking at the results for offline comparison in Tables 2 and 3 in [61], they estimated a mean global rotational error of $\sim 12^\circ \pm 6^\circ$ for the TotalCapture test dataset and $\sim 8^\circ \pm 5^\circ$ for the DIP-IMU dataset. Although [59,61] list joint angle errors, these works use SMPL as a model, while we use segment orientation directly, which may lead to some differences in comparison. In all cases, there is a moderately large standard deviation.

All the works listed used five or more sensors to predict full-body motion, whereas our work uses three sensors to predict just the upper body. It may be that the upper and lower body halves function somewhat independently in their works, and would not affect their results if they just used the upper body and pelvis sensors. Our work found a mean segment orientation error of $\sim 15^\circ$ using XSens MVN segments, and a mean of $\sim 20^\circ$ using

XSens DOT sensors for upper body inference. When we computed the joint angles (elbow and shoulder), we found mean average errors of $<4^\circ$ and standard deviations of 9–21°. These results are favorable as compared to previous works.

Furthermore, from Figures 21 and 22, we find that joint angle distributions were similar to the ground truth. However, the segment orientations had a higher margin of error. This is because joint angles were computed as the angle between two segments. If the respective segments of inference equally deviate from the respective ground truth segments, the joint angle for inference and ground truth will be theoretically the same. Thus, looking at the segment orientation error will give more insight into a model's performance.

Furthermore, we found that the forearms gave large errors with the DOT sensors but not the MVN sensors. This was confusing, since the inference was based on the sensors located on the forearms. It is likely that the forearm errors were caused by the DOT sensors drifting over time; the calibration mapping between the DOT sensors and machine learning model inputs were done once for each session, using a data frame near the beginning of the session. Thus, the DOT sensor drifting would result in errors since it would no longer match the true segment orientation. Surprisingly, the inference models seemed to be fairly immune to this drift in their estimation of the joint angles.

Overall, the DOT sensors did not perform as well as the MVN sensors. One reason for this is the imperfect mapping between the DOT sensors and the MVN system, which is what the machine learning models were trained on. The effects of the imperfect mapping are most evident when comparing the fixed mapping and the variable mapping. We found that the fixed mapping function did not perform very well at all (Figure 10 and Table 3). It turned out that the rotation matrices in Figure 10 for the individual calibrations varied substantially, with around 90° of rotation between two of them. It appears that, in general, a calibration must be performed for each individual, and again periodically over time as sensors move or drift. We note that the specific way the sensors will attach to a person's forearm will likely differ somewhat between wearers, based on arm shape and variability in sensor placement, so a universal mapping may be difficult. With the MVN system, a full calibration was performed at the beginning of each data collection session, including special poses and walking for a short distance. The MVN system benefits both from this and also the presence of sensors on all body segments, which are used to solve the full skeleton.

As described before in Section 3.3.1, the XSens MVN uses 17 IMU sensors to reconstruct full body kinematics of 23 segments. MVN sensor reference frames are located inside the sensor (Figure 8a). However, the segment reference frames have a different location than the actual sensor locations. For example, segment frames for the left forearm and right forearm are located in the respective elbow joints. In comparison, we place the forearm sensors near the wrist (Figure 8e). Thus, the linear acceleration values were different for the segment and sensor. As the input to the model, we used segment orientation and experimented with both segment acceleration and sensor acceleration. We found that when doing MVN inference, using the segment acceleration gave better results (by 0.3–0.65°). However, when doing inference for the DOT sensors, using the sensor acceleration gave much better results (by 6–7° of mean segment angular error, and 2–7° of mean joint angle error). Since our ultimate goal was a standalone system with just a few IMUs, we ultimately used the sensor accelerations as inputs to our machine learning models.

It turned out that Configuration 1 (pelvis sensor on back of pelvis) performed better than Configuration 2 (pelvis sensor on the left hip), although the results were very comparable. As seen in Figure 20, Configuration 1 had mean errors about 1° less than Configuration 2 for all of the models. It may be that the sensor location in the back of the body moves with the pelvis more closely.

Overall, the kinematics estimated by this system do provide relatively large errors, as compared to whole-body IMU-based motion capture systems. However, this system is much easier to put on and is lower cost (<500 USD). It remains to be seen if the kinematic information is sufficient for rehabilitation applications; it is promising that the overall joint angle distributions were close to the ground truth distributions, and the average joint angle

errors were small. Hopefully, with improvements, the overall trends in activity will provide insights into which upper extremity motions need additional rehabilitation.

5.3. Limitations of our Study

Motion Inference

Although we can predict upper-body motion with a reasonable error margin, there are some limitations and room for improvement in the future.

One easy way to improve the results with the DOT sensors is to increase their sampling rate. With the DOT sensors, the error increases with dynamic applications. We recorded data with the DOT sensors at a 60 Hz rate, but it would be better to record at a 120 Hz rate, which is recommended for dynamic applications.

Another limitation is that a custom calibration seems to be necessary for each person and possibly each data collection session. Since the fixed mapping with the DOT sensors did not work well, in the future, algorithms that automatically calibrate the sensor placement to a person are important to minimize the mapping error. We expect that these will need to be continuously updating algorithms that adjust even if a sensor moves over the course of a day, for example if a person takes off a wrist-mounted sensor and puts it back on again or if they disturb the orientation of the pelvis sensor. These algorithms should ideally also take into account any translational offsets between the MVN skeleton and wrist or pelvis, and thus improve the treatment of the acceleration.

6. Conclusions

In conclusion, we present several algorithms for in-home localization and kinematics reconstruction. We first present and simulate a new model-free technique for localizing a person to a region of interest. This is useful for identifying which room of a house a person is in. This technique employs a neural network to provide proximity reports based on the received RSS from beacons with known locations. Second, we validate the model-free proximity reporting by designing a neural network to localize a person to an RoI. Third, we conducted experiments validating a Bluetooth RSS fingerprinting-based approach to localization in a home environment. Finally, we presented algorithms for motion inference and data on how well three standalone IMU sensors can reconstruct the upper body kinematics. We compared two different configurations of the pelvis sensor, finding that they performed similarly. We also evaluated the possibility of a fixed mapping between the standalone IMUs and the MVN system used to train the machine learning models. We found that a calibration is necessary for each individual participant in order to get usable results. Once properly calibrated, the upper body inference system gave moderate segment orientation errors, but small mean errors for the joint angles.

It remains to be seen if the localization accuracy and the joint angle error accuracy are necessary for effective rehabilitation. It is likely important to have moderately-accurate human sensing so that the rehabilitation suggestions are based upon true data. While not explored in this paper, there may be other derived features (such as joint velocities) that may be especially useful for rehabilitation; it is unknown how well the algorithms presented here would accurately measure or estimate those. Overall, however, the work in this paper is promising for quantitative in-home rehabilitation.

Author Contributions: Conceptualization, A.K., T.R., R.M.B. and A.T.A.; data curation, A.S., D.-R.E., R.M.B. and A.T.A.; investigation, A.S. and D.-R.E.; project administration, A.K., T.R., R.M.B. and A.T.A.; software, A.S. and D.-R.E.; supervision, A.K., T.R., R.M.B. and A.T.A.; writing—original draft, all; writing—review and editing, R.M.B. and A.T.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Science Foundation (Grant # 2014499).

Institutional Review Board Statement: The study was conducted according to the guidelines of the Declaration of Helsinki and approved by the Institutional Review Board of Virginia Tech (protocol #18-877, approved 23 November 2021).

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: Links to the Virginia Tech Natural Motion Dataset and code used to train our machine learning models can be found in [60].

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Detailed List of ADL Activities

Following is a detailed list of the activities performed during data collection for the kinematics reconstruction.

Table A1. List of activities performed by the new participants. The Activity Time is the approximate number of minutes to complete the activity.

Task Group	Room Preparation and List of Activities in Details	Activity Time
(1) Walking 3 Laps in the room	**Setup/Room Preparation:** Remove any objects from the ground. Make sure there are minimal obstacles in the room while walking. **Direction/Steps of activities:** • Stay in N-pose for 10 s, then walk back and forth . • First loop: Walk slowly from one room to another room and return to the starting location. • Second loop: Walk in a medium speed from one room to another room and return to the starting location. • Third loop: Walk in a faster speed from one room to another room and return to the starting location. • Stay in N-pose for 10 s, then walk back and forth .	2.5
(2) Pick up things (clothing, books, toys, words blocks, safety pins) off floor, tidy up in order to vacuum, then vacuum.	**Setup/Room Preparation:** Place variety of items scattered across floor. Make sure vacuum is present and accessible. Initially 5 objects (book, toy, pen, cloth, coffee mug). **Direction/Steps of activities:** • Pick 2/3 items from the floor, then place items on the coffee table. • Pick remaining items from the floor, then place the items on the coffee table. • Grab the vacuum cleaner, plug it in to the power source, and start vacuuming. • Unplug the vacuum cleaner, then place it in its original location. • Pick up all the five objects (one by one) from the coffee table and place them at their designated place. For example, the book will go to the bookshelf; the coffee mug will go to the kitchen shell, etc. • Stay in N-pose for 10 s, then walk back and forth.	4
(3) Fold laundry and put it away in cabinets with appropriateabel/low drawers in multiple rooms (i.e., bedroom, linen closet, kitchen towels, hanging clothes, etc.)	**Setup/Room Preparation:** Prepare laundry basket. Include linens, hanging clothes, and folded clothes. These cloths will be placed in drawers, shelves, and linen closet labeled. **Direction/Steps of activities:** • Take the laundry basket (full of 2 shirts, 1 T-shirt, 1 pillow cover, 1 pair socks) to the linen closet • Sit in a chair. Then pick up the clothes and fold them. • After folding each clothing item, place them in their designated location. For example, T-shirts will be hung on the hanger, linens will go to the linen closet, etc. • Stay in N-pose for 10 s, then walk back and forth.	3.5
(4) Packing and unpacking a bag of groceries and put each piece in the cabinet/fridge with the appropriate label (by category).	**Setup/Room Preparation:** Need to place grocery items (5 items, e.g., a bag of coffee beans, jar of sugar, salt cellar, soda can, canned tuna) in the kitchen; also, a grocery bag should be accessible. Prepare/empty shelf space; label spots for the type of goods. **Direction/Steps of activities:** • Take the grocery bag and load all items carefully into the grocery bag. • Unpack the grocery items one by one and place them at their designated destinations. For example, the soda can will go to the fridge; the sugar jar will be placed in the kitchen cabinet. • After organizing the groceries, fold the grocery bag and put it into a drawer. • Stay in N-pose for 10 s, then walk back and forth.	2

Appendix B. Anchor Characteristics

Table A2. Anchor characteristics for proximity reporting.

Beacon ID	Position (x, y)	Parameters $(PL(d_0), \xi_u, \sigma_{s,u})$
1	(5, 37.5)	$(-57.6, -2.5, \sigma_s)$
2	(15, 37.5)	$(-68.3, -1.8, \sigma_s)$
3	(25, 37.5)	$(-68.0, -1.9, \sigma_s)$
4	(35, 37.5)	$(-60.8, -2.3, \sigma_s)$
5	(45, 37.5)	$(-60.3, -2.4, , \sigma_s)$
6	(5, 12.5)	$(-66.5, -1.9, \sigma_s)$
7	(15, 12.5)	$(-55.7, -2.6, \sigma_s)$
8	(25, 12.5)	$(-73.0, -1.2, \sigma_s)$
9	(35, 12.5)	$(-61.9, -2.0, \sigma_s)$
10	(45, 12.5)	$(-57.6, -2.6, \sigma_s)$

Appendix C. Anchor Positions and Reference Positions

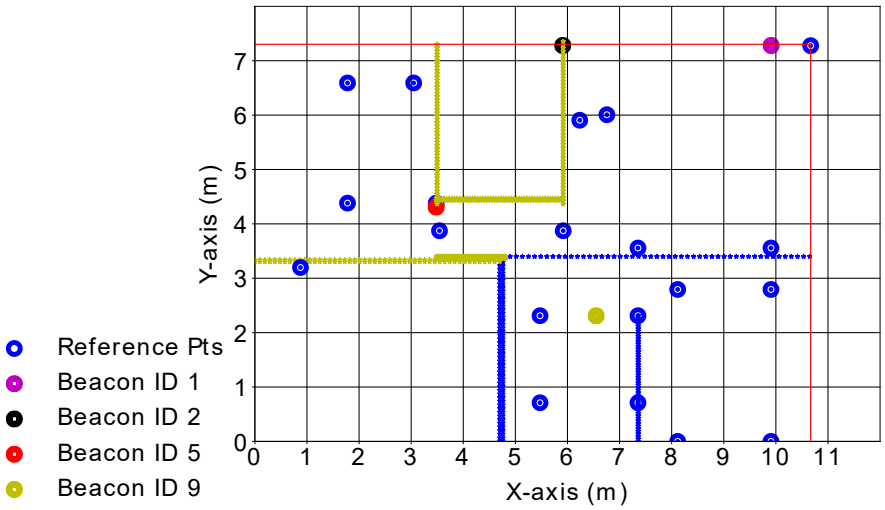


Figure A1. Positions of the anchors (beacons) used for transmitting Bluetooth signals, as well as the reference points where the signal strength was measured.

References

1. Haghi, M.; Thurow, K.; Stoll, R. Wearable devices in medical internet of things: Scientific research and commercially available devices. *Healthc. Inform. Res.* **2017**, *23*, 4–15. [CrossRef] [PubMed]
2. Pantelopoulos, A.; Bourbakis, N.G. A survey on wearable sensor-based systems for health monitoring and prognosis. *IEEE Trans. Syst. Man Cybern. Part C Appl. Rev.* **2009**, *40*, 1–12. [CrossRef]
3. Kleim, J.; Jones, T. Principles of experience-dependent neural plasticity: Implications for rehabilitation after brain damage. *J. Speech Lang. Hear. Res.* **2008**, *51*, S225–S239. [CrossRef]
4. Chen, Y.; Duff, M.; Lehrer, N.; Sundaram, H.; He, J.; Wolf, S.L.; Rikakis, T. A computational framework for quantitative evaluation of movement during rehabilitation. In Proceedings of the AIP Conference Proceedings, Toyama, Japan, 11–13 October 2011; Volume 1371, pp. 317–326.
5. Lang, C.E.; Bland, M.D.; Bailey, R.R.; Schaefer, S.Y.; Birkenmeier, R.L. Assessment of upper extremity impairment, function, and activity after stroke: Foundations for clinical decision making. *J. Hand Ther.* **2013**, *26*, 104–115. [CrossRef] [PubMed]
6. Baran, M.; Lehrer, N.; Duff, M.; Venkataraman, V.; Turaga, P.; Ingalls, T.; Rymer, W.Z.; Wolf, S.L.; Rikakis, T. Interdisciplinary concepts for design and implementation of mixed reality interactive neurorehabilitation systems for stroke. *Phys. Ther.* **2015**, *95*, 449–460. [CrossRef] [PubMed]
7. Chen, Y.; Xu, W.; Sundaram, H.; Rikakis, T.; Liu, S.M. Media adaptation framework in biofeedback system for stroke patient rehabilitation. In Proceedings of the 15th ACM international conference on Multimedia, Augsburg, Germany, 24–29 September 2007; pp. 47–57.
8. Levin, M.F.; Kleim, J.A.; Wolf, S.L. What do motor "recovery" and "compensation" mean in patients following stroke? *Neurorehabilit. Neural Repair* **2009**, *23*, 313–319. [CrossRef]

9. Slade, P.; Habib, A.; Hicks, J.L.; Delp, S.L. An Open-Source and Wearable System for Measuring 3D Human Motion in Real-Time. *IEEE Trans. Biomed. Eng.* **2022**, *69*, 678–688. [CrossRef] [PubMed]
10. Choo, C.Z.Y.; Chow, J.Y.; Komar, J. Validation of the Perception Neuron system for full-body motion capture. *PLoS ONE* **2022**, *17*, e0262730. [CrossRef]
11. Vega-Gonzalez, A.; Bain, B.J.; Dall, P.M.; Granat, M.H. Continuous monitoring of upper-limb activity in a free-living environment: A validation study. *Med. Biol. Eng. Comput.* **2007**, *45*, 947–956. [CrossRef]
12. Ambar, R.B.; Poad, H.B.M.; Ali, A.M.B.M.; Ahmad, M.S.B.; Jamil, M.M.B.A. Multi-sensor arm rehabilitation monitoring device. In Proceedings of the 2012 International Conference on Biomedical Engineering (ICoBE), Penang, Malaysia, 27–28 February 2012; pp. 424–429.
13. Stenum, J.; Cherry-Allen, K.M.; Pyles, C.O.; Reetzke, R.D.; Vignos, M.F.; Roemmich, R.T. Applications of pose estimation in human health and performance across the lifespan. *Sensors* **2021**, *21*, 7315. [CrossRef]
14. Milosevic, B.; Leardini, A.; Farella, E. Kinect and wearable inertial sensors for motor rehabilitation programs at home: State of the art and an experimental comparison. *BioMed. Eng. Online* **2020**, *19*, 25. [CrossRef] [PubMed]
15. Duff, M.; Attygalle, S.; He, J.; Rikakis, T. A portable, low-cost assessment device for reaching times. In Proceedings of the 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Vancouver, BC, Canada, 20–25 August 2008; pp. 4150–4153.
16. Uswatte, G.; Foo, W.L.; Olmstead, H.; Lopez, K.; Holand, A.; Simms, L.B. Ambulatory monitoring of arm movement using accelerometry: An objective measure of upper-extremity rehabilitation in persons with chronic stroke. *Arch. Phys. Med. Rehabil.* **2005**, *86*, 1498–1501. [CrossRef] [PubMed]
17. Michielsen, M.E.; Selles, R.W.; Stam, H.J.; Ribbers, G.M.; Bussmann, J.B. Quantifying Nonuse in Chronic Stroke Patients: A Study Into Paretic, Nonparetic, and Bimanual Upper-Limb Use in Daily Life. *Arch. Phys. Med. Rehabil.* **2012**, *93*, 1975–1981. [CrossRef] [PubMed]
18. Marschollek, M.; Becker, M.; Bauer, J.M.; Bente, P.; Dasenbrock, L.; Elbers, K.; Hein, A.; Kolb, G.; Künemund, H.; Lammel-Polchau, C.; et al. Multimodal activity monitoring for home rehabilitation of geriatric fracture patients–feasibility and acceptance of sensor systems in the GAL-NATARS study. *Inform. Health Soc. Care* **2014**, *39*, 262–271. [CrossRef] [PubMed]
19. Lemmens, R.J.; Timmermans, A.A.; Janssen-Potten, Y.J.; Smeets, R.J.; Seelen, H.A. Valid and reliable instruments for arm-hand assessment at ICF activity level in persons with hemiplegia: A systematic review. *BMC Neurol.* **2012**, *12*, 21. [CrossRef] [PubMed]
20. Bavan, L.; Surmacz, K.; Beard, D.; Mellon, S.; Rees, J. Adherence monitoring of rehabilitation exercise with inertial sensors: A clinical validation study. *Gait Posture* **2019**, *70*, 211–217. [CrossRef]
21. De, D.; Bharti, P.; Das, S.K.; Chellappan, S. Multimodal wearable sensing for fine-grained activity recognition in healthcare. *IEEE Internet Comput.* **2015**, *19*, 26–35. [CrossRef]
22. Rodrigues, M.J.; Postolache, O.; Cercas, F. Physiological and behavior monitoring systems for smart healthcare environments: A review. *Sensors* **2020**, *20*, 2186. [CrossRef] [PubMed]
23. Zhang, H.; Zhang, Z.; Gao, N.; Xiao, Y.; Meng, Z.; Li, Z. Cost-Effective Wearable Indoor Localization and Motion Analysis via the Integration of UWB and IMU. *Sensors* **2020**, *20*, 344. [CrossRef] [PubMed]
24. Paul, A.S.; Wan, E.A. RSSI-Based Indoor Localization and Tracking Using Sigma-Point Kalman Smoothers. *IEEE J. Sel. Top. Signal Process.* **2009**, *3*, 860–873. [CrossRef]
25. Zekavat, R.; Buehrer, R.M. *Handbook of Position Location: Theory, Practice and Advances*; John Wiley & Sons: Hoboken, NJ, USA, 2011; Volume 27.
26. Chen, H.; Zhang, Y.; Li, W.; Tao, X.; Zhang, P. ConFi: Convolutional Neural Networks Based Indoor Wi-Fi Localization Using Channel State Information. *IEEE Access* **2017**, *5*, 18066–18074. [CrossRef]
27. Liu, C.; Fang, D.; Yang, Z.; Jiang, H.; Chen, X.; Wang, W.; Xing, T.; Cai, L. RSS Distribution-Based Passive Localization and Its Application in Sensor Networks. *IEEE Trans. Wirel. Commun.* **2016**, *15*, 2883–2895. [CrossRef]
28. Bahl, P.; Padmanabhan, V. RADAR: An in-building RF-based user location and tracking system. In Proceedings of the Proceedings IEEE INFOCOM 2000 Conference on Computer Communications Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (Cat. No.00CH37064), Tel Aviv, Israel, 26–30 March 2000; Volume 2, pp. 775–784. [CrossRef]
29. Xie, Y.; Wang, Y.; Nallanathan, A.; Wang, L. An Improved K-Nearest-Neighbor Indoor Localization Method Based on Spearman Distance. *IEEE Signal Process.* **2016**, *23*, 351–355. [CrossRef]
30. Li, D.; Zhang, B.; Yao, Z.; Li, C. A feature scaling based k-nearest neighbor algorithm for indoor positioning system. In Proceedings of the 2014 IEEE Global Communications Conference, Austin, TX, USA, 8–12 December 2014; pp. 436–441. [CrossRef]
31. Xue, W.; Hua, X.; Li, Q.; Yu, K.; Qiu, W. Improved Neighboring Reference Points Selection Method for Wi-Fi Based Indoor Localization. *IEEE Sens. Lett.* **2018**, *2*, 1–4. [CrossRef]
32. Campos, R.S.; Lovisolo, L. A Fast Database Correlation Algorithm for Localization of Wireless Network Mobile Nodes using Coverage Prediction and Round Trip Delay. In Proceedings of the VTC Spring 2009—IEEE 69th Vehicular Technology Conference, Barcelona, Spain, 26–29 April 2009; pp. 1–5. [CrossRef]
33. Hata, M. Empirical formula for propagation loss in land mobile radio services. *IEEE Trans. Veh. Technol.* **1980**, *29*, 317–325. [CrossRef]
34. Campos, R.S.; Lovisolo, L. Mobile station location using genetic algorithm optimized radio frequency fingerprinting. In Proceedings of the ITS, International Telecommunications Symposium, Tehran, Iran, 4 December 2010; Volume 1, pp. 1–5.

35. Goldberg, D.E.; Holland, J.H. Genetic algorithms and machine learning. *Mach. Learn.* **1988**, *3*, 95–99. [CrossRef]
36. Battiti, R.; Villani, R.; Nhat, T. Neural Network Models for Intelligent Networks: Deriving the Location from Signal Patterns. In Proceedings of the First Annual Symposium on Autonomous Intelligent Networks and Systems, Los Angeles, CA, USA, 4 June 2002.
37. Xu, J.; Dai, H.; Ying, W.H. Multi-layer neural network for received signal strength-based indoor localisation. *IET Commun.* **2016**, *10*, 717–723. [CrossRef]
38. Hoang, M.T.; Yuen, B.; Dong, X.; Lu, T.; Westendorp, R.; Reddy, K. Recurrent Neural Networks for Accurate RSSI Indoor Localization. *IEEE Internet Things J.* **2019**, *6*, 10639–10651. [CrossRef]
39. Assayag, Y.; Oliveira, H.; Souto, E.; Barreto, R.; Pazzi, R. Indoor positioning system using dynamic model estimation. *Sensors* **2020**, *20*, 7003. [CrossRef]
40. Lu, C.; Uchiyama, H.; Thomas, D.; Shimada, A.; Taniguchi, R.I. Indoor positioning system based on chest-mounted IMU. *Sensors* **2019**, *19*, 420. [CrossRef]
41. Pascacio, P.; Casteleyn, S.; Torres-Sospedra, J.; Lohan, E.S.; Nurmi, J. Collaborative indoor positioning systems: A systematic review. *Sensors* **2021**, *21*, 1002. [CrossRef] [PubMed]
42. De Blasio, G.; Quesada-Arencibia, A.; García, C.R.; Molina-Gil, J.M.; Caballero-Gil, C. Study on an indoor positioning system for harsh environments based on Wi-Fi and Bluetooth low energy. *Sensors* **2017**, *17*, 1299. [CrossRef] [PubMed]
43. López-Pastor, J.A.; Ruiz-Ruiz, A.J.; García-Sánchez, A.J.; Gómez-Tornero, J.L. An Automatized Contextual Marketing System Based on a Wi-Fi Indoor Positioning System. *Sensors* **2021**, *21*, 3495. [CrossRef] [PubMed]
44. Yin, F.; Zhao, Y.; Gunnarsson, F. Proximity report triggering threshold optimization for network-based indoor positioning. In Proceedings of the 2015 18th International Conference on Information Fusion (Fusion), Washington, DC, USA, 6–9 July 2015; pp. 1061–1069.
45. Yin, F.; Zhao, Y.; Gunnarsson, F.; Gustafsson, F. Received-Signal-Strength Threshold Optimization Using Gaussian Processes. *IEEE Trans. Signal Process.* **2017**, *65*, 2164–2177. [CrossRef]
46. Bergman, N. Recursive Bayesian Estimation. Ph.D. Thesis, Department of Electrical Engineering, Linköping University, Linköping Studies in Science and Technology, Linköping, Sweden, 1999; Volume 579.
47. Patwari, N.; Hero III, A.O. Using proximity and quantized RSS for sensor localization in wireless networks. In Proceedings of the 2nd ACM International Conference on Wireless Sensor Networks and Applications, San Diego, CA, USA, 19 September 2003; pp. 20–29.
48. Pons-Moll, G.; Baak, A.; Helten, T.; Müller, M.; Seidel, H.P.; Rosenhahn, B. Multisensor-fusion for 3D full-body human motion capture. In Proceedings of the 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, San Francisco, CA, USA, 13–18 June 2010; pp. 663–670.
49. Pons-Moll, G.; Baak, A.; Gall, J.; Leal-Taixe, L.; Mueller, M.; Seidel, H.P.; Rosenhahn, B. Outdoor human motion capture using inverse kinematics and Von Mises-Fisher sampling. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6–13 November 2011; pp. 1243–1250.
50. Malleson, C.; Gilbert, A.; Trumble, M.; Collomosse, J.; Hilton, A.; Volino, M. Real-time full-body motion capture from video and IMUs. In Proceedings of the 2017 International Conference on 3D Vision (3DV), Qingdao, China, 10–12 October 2017; pp. 449–457.
51. von Marcard, T.; Henschel, R.; Black, M.J.; Rosenhahn, B.; Pons-Moll, G. Recovering accurate 3D human pose in the wild using IMUs and a moving camera. In Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018; pp. 601–617.
52. Cao, Z.; Hidalgo, G.; Simon, T.; Wei, S.E.; Sheikh, Y. OpenPose: Realtime multi-person 2D pose estimation using Part Affinity Fields. *arXiv* **2018**, arXiv:1812.08008.
53. Helten, T.; Muller, M.; Seidel, H.P.; Theobalt, C. Real-time body tracking with one depth camera and inertial sensors. In Proceedings of the IEEE International Conference on Computer Vision, Sydney, Australia, 1–8 December 2013; pp. 1105–1112.
54. Andrews, S.; Huerta, I.; Komura, T.; Sigal, L.; Mitchell, K. Real-time physics-based motion capture with sparse sensors. In Proceedings of the 13th European Conference on Visual Media Production (CVMP 2016), London, UK, 12–13 December 2016; pp. 1–10.
55. Colella, R.; Tumolo, M.R.; Sabina, S.; Leo, C.G.; Mincarone, P.; Guarino, R.; Catarinucci, L. Design of UHF RFID Sensor-Tags for the Biomechanical Analysis of Human Body Movements. *IEEE Sens. J.* **2021**, *21*, 14090–14098. 3069113. [CrossRef]
56. Schwarz, L.A.; Mateus, D.; Navab, N. Discriminative human full-body pose estimation from wearable inertial sensor data. In *3D Physiological Human Workshop*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 159–172.
57. Wouda, F.J.; Giuberti, M.; Bellusci, G.; Veltink, P.H. Estimation of full-body poses using only five inertial sensors: An eager or lazy learning approach? *Sensors* **2016**, *16*, 2138. [CrossRef]
58. von Marcard, T.; Rosenhahn, B.; Black, M.J.; Pons-Moll, G. Sparse inertial poser: Automatic 3D human pose estimation from sparse IMUs. *Comput. Graph. Forum* **2017**, *36*, 349–360. [CrossRef]
59. Huang, Y.; Kaufmann, M.; Aksan, E.; Black, M.J.; Hilliges, O.; Pons-Moll, G. Deep inertial poser: Learning to reconstruct human pose from sparse inertial measurements in real time. *ACM Trans. Graph. (TOG)* **2018**, *37*, 1–15. [CrossRef]
60. Geissinger, J.; Asbeck, A. Motion Inference Using Sparse Inertial Sensors, Self-Supervised Learning, and a New Dataset of Unscripted Human Motion. *Sensors* **2020**, *20*, 6330. [CrossRef] [PubMed]

61. Yi, X.; Zhou, Y.; Xu, F. TransPose: Real-time 3D Human Translation and Pose Estimation with Six Inertial Sensors. *arXiv* **2021**, arXiv:2105.04605.
62. Hochreiter, S.; Schmidhuber, J. Long short-term memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
63. Mahmood, N.; Ghorbani, N.; Troje, N.F.; Pons-Moll, G.; Black, M.J. AMASS: Archive of motion capture as surface shapes. *arXiv* **2019**, arXiv:1904.03278.
64. Trumble, M.; Gilbert, A.; Malleson, C.; Hilton, A.; Collomosse, J. Total Capture: 3D Human Pose Estimation Fusing Video and Inertial Sensors. In Proceedings of the British Machine Vision Conference, BMVC 2017, London, UK, 4–7 September 2017; Volume 2, p. 3.
65. Schepers, M.; Giuberti, M.; Bellusci, G. *XSens MVN: Consistent Tracking of Human Motion Using Inertial Sensing*; Xsens Technologies: Enskode, The Netherlands, 2018; pp. 1–8.
66. Roetenberg, D.; Luinge, H.; Slycke, P. *XSens MVN: Full 6DOF Human Motion Tracking Using Miniature Inertial Sensors*; Tech. Rep.; Xsens Motion Technologies BV: Enschede, The Netherlands, 2009; Volume 1.
67. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
68. Gay, W. *Raspberry Pi Hardware Reference*; Apress: New York, NY, USA, 2014.
69. Zhao, Y.; Fritsche, C.; Yin, F.; Gunnarsson, F.; Gustafsson, F. Sequential Monte Carlo Methods and Theoretical Bounds for Proximity Report Based Indoor Positioning. *IEEE Trans. Veh. Technol.* **2018**, *67*, 5372–5386. [CrossRef]
70. Geissinger, J.; Alemi, M.M.; Chang, S.E.; Asbeck, A.T. *Virginia Tech Natural Motion Dataset [Data Set]*; University Libraries, Virginia Tech: Blacksburg, VA, USA, 2020. [CrossRef]
71. Roetenberg, D.; Luinge, H.; Veltink, P. Inertial and magnetic sensing of human movement near ferromagnetic materials. In Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality, Tokyo, Japan, 10 October 2003; pp. 268–269.
72. Roetenberg, D.; Luinge, H.J.; Baten, C.T.; Veltink, P.H. Compensation of magnetic disturbances improves inertial and magnetic sensing of human body segment orientation. *IEEE Trans. Neural Syst. Rehabil. Eng.* **2005**, *13*, 395–405. [CrossRef] [PubMed]
73. Kim, S.; Nussbaum, M.A. Performance evaluation of a wearable inertial motion capture system for capturing physical exposures during manual material handling tasks. *Ergonomics* **2013**, *56*, 314–326. [CrossRef] [PubMed]
74. Al-Amri, M.; Nicholas, K.; Button, K.; Sparkes, V.; Sheeran, L.; Davies, J.L. Inertial measurement units for clinical movement analysis: Reliability and concurrent validity. *Sensors* **2018**, *18*, 719. [CrossRef] [PubMed]
75. Grassia, F.S. Practical parameterization of rotations using the exponential map. *J. Graph. Tools* **1998**, *3*, 29–48. [CrossRef]
76. Fragkiadaki, K.; Levine, S.; Felsen, P.; Malik, J. Recurrent network models for human dynamics. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 4346–4354.
77. Jain, A.; Zamir, A.R.; Savarese, S.; Saxena, A. Structural-RNN: Deep Learning on spatio-temporal graphs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 5308–5317.
78. Martinez, J.; Black, M.J.; Romero, J. On human motion prediction using recurrent neural networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 2891–2900.
79. Taylor, G.W.; Hinton, G.E.; Roweis, S.T. Modeling human motion using binary latent variables. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 3–6 December 2007; pp. 1345–1352.
80. Ionescu, C.; Papava, D.; Olaru, V.; Sminchisescu, C. Human3.6M: Large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE Trans. Pattern Anal. Mach. Intell.* **2014**, *36*, 1325–1339.
81. Pavllo, D.; Grangier, D.; Auli, M. Quaternet: A quaternion-based recurrent model for human motion. *arXiv* **2018**, arXiv:1805.06485.
82. Sutskever, I.; Vinyals, O.; Le, Q.V. Sequence to sequence learning with neural networks. In Proceedings of the Advances in Neural Information Processing Systems, Montreal, QC, USA, 8–13 December 2014; pp. 3104–3112.
83. Chung, J.; Gulcehre, C.; Cho, K.; Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv* **2014**, arXiv:1412.3555.
84. Schuster, M.; Paliwal, K.K. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.* **1997**, *45*, 2673–2681. [CrossRef]
85. Bahdanau, D.; Cho, K.; Bengio, Y. Neural machine translation by jointly learning to align and translate. *arXiv* **2014**, arXiv:1409.0473.
86. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
87. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training. 2018. Available online: https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf (accessed on 25 April 2020).
88. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv* **2018**, arXiv:1810.04805.
89. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI Blog* **2019**, *1*, 9.
90. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv* **2019**, arXiv:1910.10683.
91. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language models are few-shot learners. *arXiv* **2020**, arXiv:2005.14165.

92. Rush, A.M. The annotated transformer. In Proceedings of the Workshop for NLP Open Source Software (NLP-OSS), Melbourne, Australia, 20 July 2018; pp. 52–60.
93. Alammar, J. The Illustrated Transformer. 2018. Available online: http://jalammar.github.io/illustrated-transformer (accessed on 25 April 2020).
94. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An imperative style, high-performance deep learning library. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 8024–8035.
95. Markley, L.; Cheng, Y.; Crassidis, J.; Oshman, Y. Averaging Quaternions. *J. Guid. Control Dyn.* **2007**, *30*, 1193–1196. [CrossRef]
96. Yin, K.; Pai, D.K. Footsee: An interactive animation system. In Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, San Diego, CA, USA, 26–27 July 2003; pp. 329–338.
97. Fang, S.H.; Lin, T.N. Indoor location system based on discriminant-adaptive neural network in IEEE 802.11 environments. *IEEE Trans. Neural Netw.* **2008**, *19*, 1973–1978. [CrossRef] [PubMed]
98. Loper, M.; Mahmood, N.; Romero, J.; Pons-Moll, G.; Black, M.J. SMPL: A skinned multi-person linear model. *ACM Trans. Graph. (TOG)* **2015**, *34*, 248. [CrossRef]