Step detection using LSTM networks

Stef Vandermeeren[1], Herwig Bruneel[2], Heidi Steendam[1]
[1] TELIN/IMEC
[2] TELIN
Ghent University, Belgium,
e-mail: {firstname}.{lastname}@ugent.be

***Abstract*—During the last years, a lot of research has been carried out on indoor localisation techniques. These techniques can provide valuable information to a very broad range of applications, e.g. in situations where a user needs to find his/her way inside a large building like a shopping centre or an airport. One popular approach to track the location of a user is to use an inertial measurement unit (IMU), which is the combination of an accelerometer, gyroscope and sometimes magnetometer. The IMU allows us to track a user with a people dead reckoning (PDR) system by detecting each step the user takes, combined with the length and heading of that step. To this end, an algorithm must first be able to recognise the boundaries in the signal corresponding to a single step. In this paper, we use the accelerometer to detect the start and end of each step using a neural network based on the LSTM architecture. This resulted in a precision of** $98.8\%(99.5\%)$ **and a recall of** $99.2\%(99.3\%)$ **for detecting the start(end) of a step.**

I. Motivation

In the literature, several methods can be found to detect the steps of a user. A first method [1], employs the periodic pattern in the measured acceleration. An example is shown in Figure 1, where a user took four steps with a handheld smartphone. A drawback of the algorithms in [1] is that they all have some parameters that need to be tuned to the specific user, which limits their usefulness. In [2], the authors use an artificial neural network (ANN) to count the number of steps a user takes. To this end, the authors divide the acceleration into shorter fragments. The ANN is then trained to output 1 when the fragment corresponds to a step and 0 otherwise. A problem with this approach is that more than one step can be present in the fragment. In [3], the authors use Bidirectional Long Short-Term Memory Recurrent Neural Networks (BLSTM-RNNs) to count steps. BLSTM-RNNs are a type of neural network that can handle time series with an unknown size, and can also use the information of the time series to predict the output at other time steps. Hence, these networks are capable to show some kind of memory. As the authors use a bidirectional network, the neural network is able to use information from previous and future time steps of the input time series. Common to all these works is that the authors do not verify that each step is detected at the right time, which is important if the step length is determined from the acceleration samples of one step. In our previous work [4], we developed a method to estimate the step length using an accelerometer only. In that work, we manually divided the measured acceleration into smaller fragments that corresponded to a step. However, to be practically useful we also need to be able to automatically subdivide the measured acceleration signal into steps. In this work, we develop a deep learning-based step detection algorithm, which determines the start and end of each step, and we also verify how accurately our algorithm can estimate the start and end. The reason that we determine the start and end of each step separately, and not just the boundaries of a step, is to prevent that if there is a pause between two steps, we would consider this pause as a step.

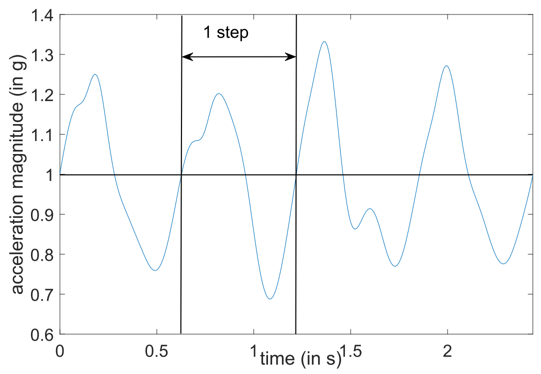

Figure 1. Filtered acceleration magnitude for a user that took 4 steps with a handheld phone

II. Start and end detection of a step

In this section, we describe how we determine the start and end of each step from the measured acceleration (sampled at 100Hz) of a handheld phone in texting position. In Figure 1, we show the filtered magnitude of the measured acceleration when a user took four steps, and the boundaries for one step. From this figure, we can easily recognize the four steps; the boundaries of each step can be found by detecting the instants where the acceleration magnitude crosses 1g with a positive slope. Hence, in theory, the step boundaries can be easily found by determining the instants where the acceleration magnitude crosses 1g. However, noise and acceleration not related to taking steps (e.g. shaking the IMU), can result in multiple positive crossings of 1g within a single step. In this work we use a Long Short-Term Memory (LSTM) network [5], which is a recurrent neural network (RNN), to determine the start and end of each step. An LSTM network is a type of neural network that is able to detect temporal patterns in a data sequence. While regular neural networks start from a fixed-length input and transform this input into a hidden state and finally into a fixed-length output; recurrent neural networks, however, make it possible to also use the hidden state of previous inputs, enabling LSTMs to have some kind of memory. This makes LSTMs ideally suited to make predictions based on data from time series (in our case the acceleration signal). In this section, we only describe how the start of a step can be determined. The process to determine the end of a step, however, is very similar. In Figure 2, we show the architecture of our LSTM network. Each time a new sample of the accelerometer is available, we provide the x-,y- and z-component of the acceleration together with the magnitude, i.e. a four-dimensional vector, as an input to the LSTM network. Next, this input is fed to two LSTM layers with 10 hidden units each. The optimal value for this parameter was found by simulation. Finally, we transform the output of the final LSTM layer into the desired output using a dense layer. Before we can start training our LSTM, we first need to define the desired output of the LSTM. Ideally the output of the LSTM network is equal to one when a step begins and zero elsewhere. However, this would complicate the training of the LSTM network as a model that detects the start of a step only one sample late is considered equally bad as a model that did not detect the start of that step. To solve this problem, we make the output equal to 1 in an interval around the start of a step. A final problem is that the LSTM network cannot decide about the start of a step based on only the first acceleration sample of that step, but needs to observe the acceleration for a short interval of the signal. To solve this problem, we add a delay to the correct output, i.e. the training output will become one for the start of a step after this delay. As long as this delay is shorter than a step, we are still capable to estimate the step boundaries in real time. Once we have a trained model, we can input the accelerometer data sample by sample to test the model on new data. The procedure to determine the end of a step is very similar to the procedure we discussed in this section to determine the start of a step. The only difference is that to determine the end of a step, we do not add a delay to the output as only previous data suffices to predict if a sample corresponds to the end of a step.

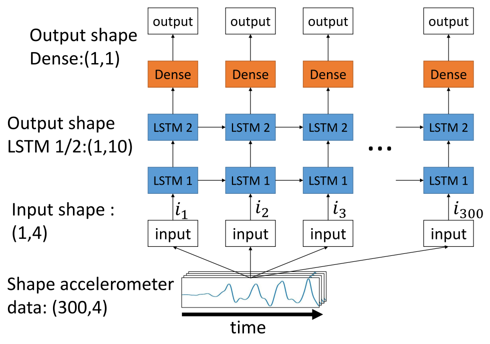

Figure 2. Architecture of our LSTM network for detecting the start(end) of a step

III. Results

To train and test our step detection algorithm, we gathered acceleration data for in total 6226 steps from which 3120 were used to train the algorithm, and 3106 for testing. In this section, we evaluate if our step detector is capable to detect the start and end of steps, and if these are also detected at the correct instants. We assume that the start(end) of a step is detected correctly, if the time between the detected and true start(end) of a step is less than $0.25s$ (optimal value found by simulation). To evaluate our algorithm, we determine the recall and precision for both the estimated start and end of steps and both for the training and test set, i.e. precision $= \frac{tp}{tp+fp}$ and recall $= \frac{tp}{tp+fn}$, where tp is the number of starts(ends) of a step that are predicted correctly, i.e. are less than $0.25s$ from a true start(end), fp is the number of starts(ends) that are predicted at a wrong time (difference with a true start(end) larger than $0.25s$), and fn is the number of true starts(end) of a step that are not detected by our detection algorithm. Hence, precision tells us if our detector detects a start(end) of a step, how likely it indeed is a start(end), while recall tells us if a start(end) of a step occurred, how likely our algorithm will detect this. In Table I, we give the recall and precision for the detection of the start and end of a step both on the training and test set. For detecting the start of a step, we achieved a recall and precision of respectively 99.2% and 98.8% on the test set, while for detecting the end of a step, we achieved respectively a recall of 99.3% and a precision of 99.5%. From this table, we can see that in general, our algorithm has slightly higher recall than precision. We also notice that our algorithm is slightly better at detecting the end than the start of a step. A reason for this can be that to correctly detect the start of a step, we needed to add a delay to the output of our LSTM network, which makes the detection of the start of a step more complicated.

Table I
RECALL AND PRECISION FOR DETECTION OF START(END) OF A STEP ON TRAINING AND TEST SET

	start of steps		**end of steps**	
	training set	**test set**	**training set**	**test set**
recall(%)	98.9	99.2	99.3	99.3
precision(%)	98.5	98.8	98.6	99.5

IV. Conclusions

In this paper, we designed a step detection algorithm that is able to detect the start and end of each step. To this end, we used an LSTM network that is able to take into account the temporal patterns in the acceleration signal. We also verified that the starts and ends of a step were detected at the correct instant. Our approach resulted in a precision and recall around 99% for both detecting the start and end of a step. In our future work, we can use the start and end of step to e.g. estimate the length and/or orientation of each step.

Acknowledgement

This work is funded by the EOS grant 30452698 from the Belgian Research Councils FWO and FNRS.

References

[1] A. Brajdic and R. Harle. Walk detection and step counting on unconstrained smartphones. In *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, pages 225–234. ACM, 2013.
[2] J. Kupke, T. Willemsen, F. Keller, and H. Sternberg. Development of a step counter based on artificial neural networks. *Journal of Location Based Services*, 10(3):161–177, 2016.
[3] Marcus Edel and Enrico Köppe. An advanced method for pedestrian dead reckoning using blstm-rnns. In *Indoor Positioning and Indoor Navigation (IPIN), 2015 International Conference on*, pages 1–6. IEEE, 2015.
[4] Stef Vandermeeren, Samuel Van de Velde, Herwig Bruneel, and Heidi Steendam. A feature ranking and selection algorithm for machine learning-based step counters. *IEEE Sensors Journal*, 18(8):3255–3265.
[5] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to forget: Continual prediction with lstm. 1999.