Hacking Rokoko Smart Gloves for Tool Detection using Recurrent Neural Networks

Zhouyao Yu[1,2], Ruben Schlonsak[1,3], and Denys J.C. Matthies[1,3]

[1] Technical University of Applied Sciences Lübeck, Germany
[2] East China University of Science and Technology, Shanghai, China
[3] Fraunhofer IMTE, Lübeck, Germany

Abstract. The prolonged use of tools emitting vibration may cause damage in fingers and hands, known as the Raynaud / White Finger Syndrome. A solution is monitoring the device use for those who work long hours with heavy tools. In this paper, we demonstrate how to hack the Rokoko smart gloves to enable them to detect and monitor the tool in use. We demonstrate how to capture data and how to train a machine learning model. The data collected stems from six 9-DoF IMUs that are placed at each finger and the back of the hand. The model is a Recurrent Neural Network (RNN) with a Long Short-Term Memory (LSTM) layer. In this work-in-progress we show the exemplary discrimination of idle from the use of scissors, knife, and hammer with a precision of 70.75%. As workers already wear gloves, we believe commercial smart gloves to be highly practical for similar applications.

Keywords: Smart Glove, Neural Networks, RRN, LSTM, IMU, Gesture Recognition, Tool Detection

1 Introduction

Prolonged use of heavy tools, such as jackhammer, can lead to nerve damage in fingers and hands. According to research on jackhammer drillers, the longer they use the heavy tools the more they suffer from hand-arm vibration syndrome (HAVS) also called Raynaud or White Finger Syndrome [1]. A monitoring device for tool use is necessary to protect the health of the worker [2]. The current state in tool recognition demonstrates that significant research exists concerning tool recognition or worker activities recognition via computer vision or sensor technology [3]. However it requires extra work to mount cameras in daily work situation, which is a inconvenience. For tracking hand-arm vibration, monitoring the device use can also be done with sensors attached to the drill, but which includes obstructing wires or additional weight from a battery [4].

On the other hand, workers usually wear protection gloves when operating such devices. Therefore, utilizing a smart glove can be promising and practical approach. Once the tool is known to our system, we can easily calculate the daily doses of vibration based on the HAV intensity ratio, which is usually reported from the tool's datasheet. To demonstrate the feasibility of a tool detection with

Fig. 1. Typical tools that are used in a various kinds of shop floors may include a masher, grinder, hammer, angle grinder, and many more. All these tools emit considerable hand-arm vibration when used, which is unhealthy and should be tracked by the worker as demanded by legal regulations in European countries such as Germany.

smart gloves, we hacked the Rokoko smart gloves. In this research, we focus on the data acquisition and final tool detection. For the tool detection, an recurrent neural network containing LSTM layer is used. Data are collected through glove's mounted IMUs, being normalized, and then feed to a NN.

In accrodance to Wobbrock [5], this research is considered an artifact contribution that proposes a feasible and low-cost solution for a working environment tool usage monitoring.

2 Related Work

Plenty of works related to IMU based gesture recognition and tool recognition area have long been conduct in past years, among which there are some inspiring work. The detection used to rely on conventional Bayesian algorithms and state machines, while currently Neural Networks are considered the state-of-the-art for high precision recognition with multi-dimensional data.

Plenty of works have addressed IMU-based gesture and tool recognition. Early approaches relied on Bayesian models and state machines, while neural networks now dominate for high-precision recognition of multidimensional data. Numerous studies have advanced automatic activity recognition across domains, with related contributions from computer-assisted intervention (CAI).

Optical Surgical Workflow Recognition. Sahu et al. [3] proposed tool and surgical phase recognition using CNN features and random forests on the M2CAI16 dataset, achieving ∼50% accuracy. While effective in the OR, camera-based solutions are costly and impractical for daily environments with limited visibility.

Attaching Wearable Sensors. Fort et al. [6] suggested embedding accelerometers on the body or tools to monitor hand–arm vibration (HAV) during use of drills, jackhammers, and similar tools, demonstrating feasibility of embedded ML on low-power devices.

Hand–Arm Vibration Exposure in Rock Drill Workers. Clemm et al. [7] compared hand- vs. tool-mounted accelerometers, finding that hand-based sensors may underestimate HAV depending on grip, highlighting limitations of sensor placement.

HAV Estimation with Smartwatches. Matthies et al. [8, 9] proposed detecting tool type with smartwatch sensors to estimate HAV dose using manufacturer vibration ratings. They showed that built-in microphones and accelerometers could distinguish tools such as hammer drills, jigsaws, and manual hammers.

Wristband for Activity and Tool Detection. Tao et al. [10] combined IMU and sEMG data via a Myo armband and CNN, achieving up to 98% accuracy for tasks like screwdriver turning or hammering. However, slippage and discomfort of wearable devices limit practical use.

Recognition with Neural Networks. Koch et al. [11] demonstrated that RNNs outperform traditional methods in gesture recognition with shorter sample windows, while LSTM layers enhance robustness for longer sequences. Rivera et al. [12] further showed LSTM-based models achieve >80% accuracy in daily activity recognition, suggesting strong suitability for IMU-based tool use recognition.

3 Smart Glove

3.1 Commercial Hardware

The hardware used are the Rokoko smart gloves[4]. On each glove, there are six 9-axis IMUs. Each finger features a single IMU while the sixth is attached to the back of the hand. By the hybrid IMU and EMF fusion technology, it can support high accuracy and frequency that up to 100 FPS. In addition, the wireless communication module enables 100 meters tracking range, which really fits the labour environment.

As a commercial product, the interface provided to access smart gloves is looks quite sophisticated. Within their software, called Rokoko studio, some black-box algorithms calculate the sensor data and display the tracked hand by a 3d avatar (ses figure 6). The software provides a comprehensive platform that allows for device connectivity, real-time gesture representation, streaming capabilities, and animation recording, we are unable to access the raw data.

However, the manufacturer does not currently allow the developer to directly acquire raw sensor data. Thus, the ultimate crucial stage in the development process is to obtain raw sensor data to allow for a more detailed analysis and a customised NN model training.

3.2 Hacking the Glove

Hacking the software seems quite impossible due to the signed packages and many other unknown factors. So, we decided to attack at a different level. Our starting point is the Rokoko Studio after establishing a wireless network connection to the smart gloves. Notably, it becomes feasible to capture data packets through network packet analysis. By using the tool Wireshark, we can sniff data

[4] Rokoko: Capture accurate finger motions with Smartgloves (`https://www.rokoko.com/products/smartgloves`)

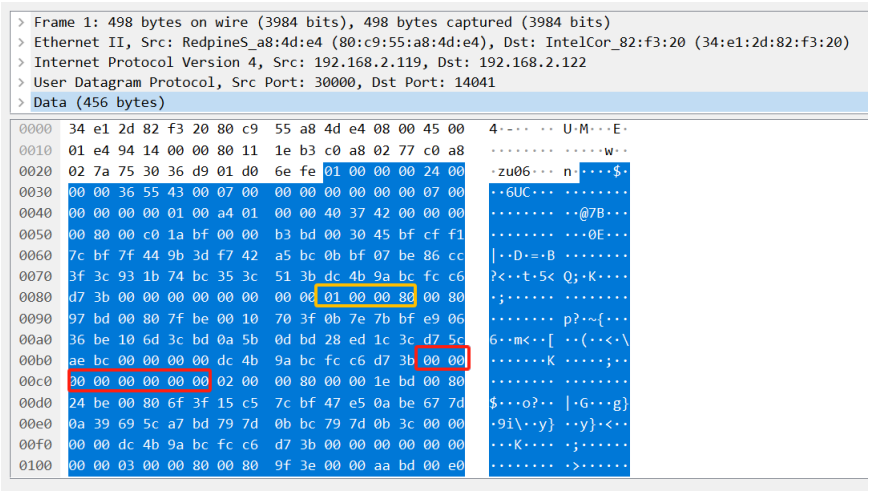

Fig. 2. Rokoko glove data package: Payload is highlighted in blue. Heading code in yellow box. Ending zeros in red box.

packages with a 456-byte payload, which are constantly transmitting from gloves by UDP protocol to the computer [13].

Figure 2 shows the payload highlighted in blue, which is characterized with a split by a series of zeros. Since each glove has six IMUs with the same data format, the payload is apparently divided into six similar blocks, starting with sensor heading code and ending with eight zeros. After rearrange data blocks (figure 3), a same row of data is ignored because seldom do two sensors read the same, especially in a resolution of 64 bits. For some most IMUs, we collect data from the accelerometer, gyrometer, magnetometer, and an additional temperature sensor value, which is also embedded [14, 15]. Compared to the valid data on the left, it is easy to identify all 10 readings, each consisting of 32 bits.

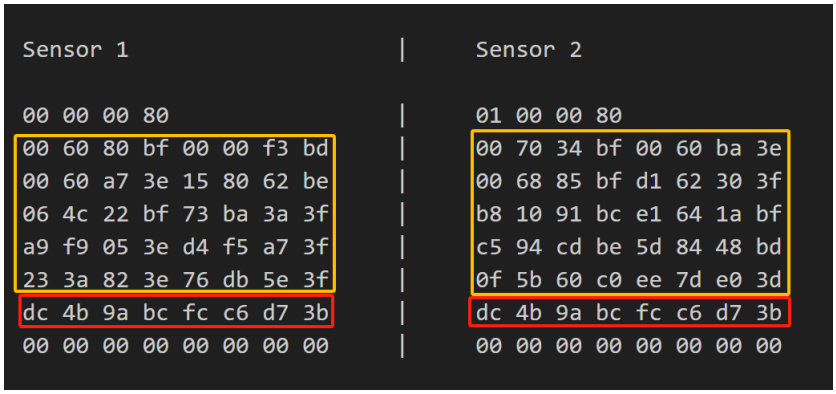

Fig. 3. Data block of 2 IMUs: Ignored data marked in red box. Valid data in yellow box

Finally, a series of motion tests is conducted to identify the id of the IMU. The readings for each sensor and the sequence of bytes present the actual data. In conclusion, each network package composes a header and six IMU data. For each imu, it contains 60 hexdecimal data, starting with IMU sequence number, then following up with a 32-bit accelerometer reading, a 32-bit gyrometer reading, a 32-bit magnetometer reading, a thermometer reading, and finally surrounded by filling zeros.

4 Machine Learning

4.1 Data Collection

The basis for a machine learning approach is to train a model based on a data set. To establish the data set, we selected four classes, including three commonly used tools, which are scissors, knife and a hammer. The fourth class is the default class containg various other gestures as well as resting motion. The selected motions for testing involve cutting with a knife and scissors, as well as hitting with a hammer. Apparently one is enable to select other motions and devices from the shop floor, such as a drill, grinder etc. However, as this is a work-in-progress and due to practicability, we selected these ones.

During the sampling process, several subjects are tasked to perform these actions while facing various directions and adopting different angles. This deliberate variation in orientation serves the purpose of preventing overfitting during the subsequent model training phase. By incorporating diverse perspectives and angles during sampling, the model can develop a more robust understanding of the targeted motions, enhancing its ability to generalize effectively to various real-world scenarios.

For the purpose of raw data acquisition to support live recognition, the Scapy library has been employed to implement network data capture functionality, seamlessly integrated into the recognition system. Through this library, the script is empowered to capture network packages originating from a specified IP source, while concurrently applying a defined message filter [16]. This integration facilitates real-time data collection, allowing the recognition system to continuously receive and process relevant information for accurate and dynamic motion analysis.

4.2 Signal Processing

Signal Cleaning and Filtering After the data acquisition, the data is still ambiguous to be used, because of unstable frame rate and soiled readings. The frame rate problem is caused by mismatch of sensors and throughput capacity of internet. For each IMU, there is 3-axis gyrometer and accelerometer, which work max to 100Hz. Moreover, the magnetometer works in lower frequency. However, the incoming data is more frequent than magnetometer updates, which results in a mismatch of readings between magnetometer and current position. Further,

due to extra-high frequency, the incoming data sometimes get stuck and are not distributed equably in time. The solution is to lower the frequency and add a validation check in code. The data is refined to 20Hz update rate. With the help of flag bit in acceleration reading, it is possible to detect whether a frame containing magneto reading. Only the frame with full information can be viewed valid.

For soiled reading, as is mentioned before, excluding thermometer, there are 54 data in one frame. A frame can be viewed as usable only if all readings are valid, which is difficult. During test, phenomenon is common that less than 5 frames are usable in one second that cannot match the requirement of recognition. The solution is to fill the spoiled datum by the former. It is considered not normal if a single datum suddenly goes to zero, given the sample rate is set relatively high. When a new frame is received, it will be compared with two frames of data that are temporally close to each other. If it is invalid, it will be filled by former data.

Extra Data Column According to the feature of accelerometer, the impact of gravity always reflect on the readings. Take using scissors for instance, the activities of horizontal cutting and vertical cutting will be viewed as two different actions because z-axis is added by gravity reading in former cases and x-axis is the same in the latter case. To decrease the need of sample amount, there are a new column of data added to each IMU, which is the 3d vector norm of acceleration calculated by: $norm = \sqrt{acc_x^2 + acc_y^2 + acc_z^2}$. Thus only the change of acceleration is record, and the data looks like figure. After going through all processing steps, we end up with sequences of 40 samples that reflecting motion in 2 seconds, which is our chosen window size to feed the neural network.

4.3 Recurrent Neural Network and Long Short-term memory

Given the temporal sequence of hand movement, a recurrent neural network is considered better in handling this kind of inputs following related work (see

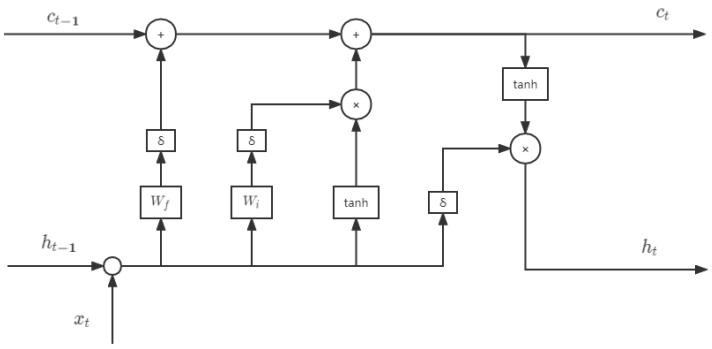

Fig. 4. Our approach of the LSTM cell.

above). In order to get more accurate recognition result, a larger window of input will be better to avoid over-fitting. In this case, LSTM cells are applied in the network as follows:

As is shown in figure 4. The memory of context is h_t, with the use of sigmoid function, the network is able to control the memorization and forgetting of context-specific information, and thus the LSTM cell enables the network to process a larger sequential input.

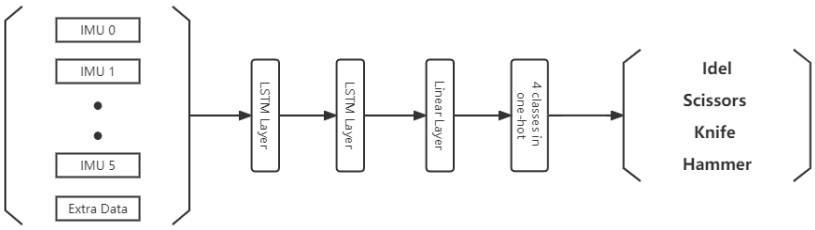

Fig. 5. The Network flow chart of our network layer design used.

The network used in this project consist of two LSTM layers and one linear layers. As is showed in figure 5, the input includes the filtered raw data of every imus and some extra data that is calculated according to the raw ones. The the input will go through 2 LSTM layers which are fully connected. Then a linear layer will flatten the result as the weight of each tool class.

4.4 Model Training

As previously stated, the algorithm to classify is a RNN. Compared to other classifiers, it is more suitable to handle input in time sequence, and thus pave the way for possible real-time detection. Given the input number of each sample is 40, the LSTM is chosen. Different from traditional RNNs, with its context cell, a LSTM model can handle larger input sample in time sequence. The label of data is encoded in one-hot. In this network, the input data is be fed to a double layers LSTM with a hidden size of 36. The outcome of the sequential input is put into a full connection layer and then the data is flattened to the size of label.

The dataset used to train our model contains five users performing each activity for at least a minute. Our series of data recordings therefore contain different execution styles and thus we do not expect a very high precision. Each series of data match a tool.

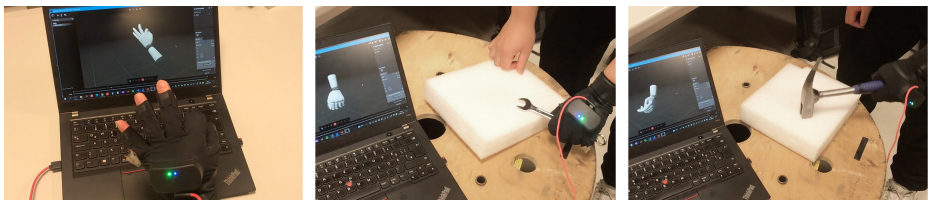

Fig. 6. Showing the Rokoko Studio's visualisation with different gestural activites.

5 Evaluation

The test data is generated in the same way as the training data. For our test set, we collected data from a single user and so to speak performed a leave-one-user out to also gain an understanding of the interpersonal feasibility of the trained model. Figure 7 shows the result of test, where the label 0,1,2,3 refer to idle, scissors, knife and hammer respectively. In the test, each tool is used from 10 to 20 seconds. Overall we achieve a precision of 70.75%, which is fair given the rather small training and test set. As shown, the use of knife is relatively accurate because the execution style did not greatly differ for this activity across users. In this case the motion of using knife is on one platform so it is easy to recognize through the norm of acceleration. To improve the accuracy of other tools, more information should be added. Since the dataset is relatively small for training a stable NN, the model may still overfit to specific users. It is to assume that greater volume of training and test data will most likely result in a substantial boost of accuracy.

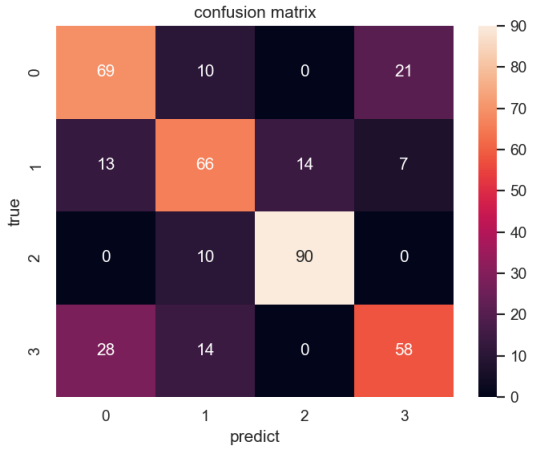

Fig. 7. Confusion matrix generated from our evaluation (Label description: 0 = idle, 1 = scissors, 2 = knife, 3 = hammer)

6 Conclusion and Future Work

In this paper, a tool detection application was developed based on the Rokoko smart gloves. For classification algorithm, a recurrent neural network with two long short-term-memory layers was applied to recognize four classes. Distinguishing the idle state from using scissors, a knife and a hammer resulted in a model precision of 70.75%. The performance looks promising, but should be improved with greater samples in future. Also, applying the glove in a shop floor and training a model with commonly used tools is the next step. Utilizing a commercial smart glove for identifying the hand-arm vibration exposure dosage is a promising approach and should be pursued in future.

References

1. A. K. Dasgupta and J. Harrison, "Effects of Vibration on the Hand-Arm System of Miners in India," *Occupational Medicine*, vol. 46, no. 1, pp. 71–78, 02 1996.
2. A. d. B. zum Vibrationsschutz, "Erstmals grenzwerte für die vibrationsbelastung," 2007.
3. M. Sahu, A. Mukhopadhyay, A. Szengel, and S. Zachow, "Tool and phase recognition using contextual cnn features," *arXiv preprint arXiv:1610.08854*, 2016.
4. J. Kuczyński, "Improved methods of assessment of vibration risk," 2014.
5. J. O. Wobbrock and J. A. Kientz, "Research contributions in human-computer interaction," *interactions*, vol. 23, no. 3, pp. 38–44, 2016.
6. A. Fort, E. Landi, R. Moretti, L. Parri, G. Peruzzi, and A. Pozzebon, "Hand-arm vibration monitoring via embedded machine learning on low power wearable devices," in *IEEE International Symposium on Measurements & Networking (M&N)*. IEEE, 2022, pp. 1–6.
7. T. Clemm, K.-C. Nordby, L.-K. Lunde, B. Ulvestad, and M. Bråtveit, "Hand-arm vibration exposure in rock drill workers: A comparison between measurements with hand-attached and tool-attached accelerometers," *Annals of Work Exposures and Health*, vol. 65, no. 9, pp. 1123–1132, 2021.
8. D. J. Matthies, M. Haescher, G. Bieber, and S. Nanayakkara, "Hand-arm vibration estimation using a commercial smartwatch," *14th International Conference on Hand-Arm-Vibration*, 2019.
9. D. J. Matthies, G. Bieber, and U. Kaulbars, "Agis: automated tool detection & hand-arm vibration estimation using an unmodified smartwatch," in *Proceedings of the 3rd International Workshop on Sensor-based Activity Recognition and Interaction*, 2016, pp. 1–4.
10. W. Tao, Z.-H. Lai, M. C. Leu, and Z. Yin, "Worker activity recognition in smart manufacturing using imu and semg signals with convolutional neural networks," *Procedia Manufacturing*, vol. 26, pp. 1159–1166, 2018, 46th SME North American Manufacturing Research Conference, NAMRC 46, Texas, USA.
11. P. Koch, M. Dreier, M. Böhme, M. Maass, H. Phan, and A. Mertins, "Inhomogeneously stacked rnn for recognizing hand gestures from magnetometer data," in *2019 27th European Signal Processing Conference (EUSIPCO)*. IEEE, 2019, pp. 1–5.
12. P. Rivera, E. Valarezo, M.-T. Choi, and T.-S. Kim, "Recognition of human hand activities based on a single wrist imu using recurrent neural networks," *Int. J. Pharma Med. Biol. Sci*, vol. 6, no. 4, pp. 114–118, 2017.
13. J. Beale, A. Orebaugh, and G. Ramirez, *Wireshark & Ethereal Network Protocol Analyzer Toolkit*. Elsevier, Dec. 2006.
14. R. Chandrasiri, N. Abhayasinghe, and I. Murray, "Bluetooth Embedded Inertial Measurement Unit for Real-Time Data Collection for Gait Analysis," *International Conference on Indoor Positioning and Indoor Navigation*, 2013.
15. A. P. Moschevikin, A. Sikora, P. V. Lunkov, A. A. Fedorov, and E. I. Maslennikov, "Hardware and software architecture of multi mems sensor inertial module," in *2017 24th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*. Saint Petersburg, Russia: IEEE, May 2017, pp. 1–3.
16. S. Bansal and N. Bansal, "Scapy-a python tool for security testing," *Journal of CS & SB*, vol. 8, no. 3, p. 140, 2015.