A Platform and Methodology Enabling Real-Time Motion Pattern Recognition on Low-Power Smart Devices

Omid Sarbishei
Research and Development Department, Motsai Research, Saint Bruno, QC, Canada
o.sarbishei@motsai.com

***Abstract*— This article presents a low-power platform, i.e., Neblina™ system-on-module, with extensive hardware variants targeting Internet of Things applications. The base hardware utilizes a sensor fusion algorithm for 3D orientation tracking alongside a novel configurable framework for real-time Motion Pattern Recognition (MPR) on ARM Cortex M4F using inertial sensors. The MPR engine consists of configurable blocks performing shock-aware segmentation, histogram feature extraction, and classification using a single-hidden-layer Feedforward Neural Network. The framework can be used for human fitness/daily activity tracking or shock pattern recognition, e.g., in sports, such as tennis, golf, hockey, etc. Our platform can deliver multimodal user feedback as well. Experimental results have evaluated Neblina's MPR framework in terms of A) accuracy in human fitness activity recognition using some existing datasets, B) memory usage and latency for real-time execution on Cortex M4F and C) low power consumption for a longer lasting battery.**

I. INTRODUCTION

Motion capture and Motion Pattern Recognition (MPR) using low-cost Inertial Measurement Units (IMU) including 3-axis accelerometers and 3-axis gyroscopes have vast applications in consumer wearable and industrial markets. In essence, low-power smart connected objects, which involve IMUs, can leverage MPR for 1) human fitness activity recognition [4, 6, 8], 2) human daily activity tracking [3, 9], 3) qualitative motion analysis in sports, e.g., golf swings [10], or 4) shock pattern recognition in industrial domains, e.g., to analyze road conditions based on the IMU data from a moving vehicle [11]. User adoption for such devices sometimes necessitates providing multimodal feedback signals including visual, audio, and (or) vibro-tactile [12].

While numerous solutions exist in the literature to perform application-specific MPR using IMUs [3-4, 8], they mostly perform a filtering/calibration pre-processing step followed by 1) *segmenting* sensor data streams, 2) extracting/selecting *characteristics*, and 3) *classification*, i.e., mapping segments to human daily activities, sports activities, correct/incorrect execution of an exercise, abnormal shock patterns, etc.

The most common approach for segmentation is the use of an overlapping sliding window [8]. The sliding window length and its step-size can be selected based on the biomechanics of the motion involved in the target activities. As a rule of thumb, the window-length is required to be long enough to cover at least one cycle of the slowest target activity/pattern, while the step size for the sliding window should be small enough to cover the high-frequency movements of interest. For instance, the RecoFit methodology proposed by [8] has chosen a 5s window sliding at 200ms steps for human fitness activity recognition, and that has shown to be effective considering the involvement of 94 subjects and 26 activities [8].

The approach in [8] explores a plethora of features alongside a linear Support Vector Machine (SVM) classifier for activity recognition. Linear SVM generally performs well alongside a rich feature set. SVM classifiers with nonlinear kernel functions have also been explored in the literature for activity recognition [3]. However, the computations required for training will grow quadratically with respect to the number of datapoints. This limits their applicability for large datasets.

The methodology discussed in [3] has evaluated a pool of 293 classifiers including K-Nearest Neighbors (KNN), Feedforward Neural Networks (FNN), Ensemble Methods, SVMs, etc., to select the most efficient classifier for human daily activities. The solutions in [3, 4] have both concluded that KNN outperforms other classifiers for such problems. However, from a practical point of view, KNNs require major memory usage and suffer from long latency, when dealing with large datasets. This limits their applicability for real-time execution on a low-power and low-cost microcontroller, such as ARM Cortex M4F. Alternative solutions, such as FNNs are as capable as KNNs in terms of building accurate classifiers, when A) They follow efficient segmentation and feature selection pre-processing steps and B) Enough engineering effort and time is put into training them [7]. Pre-trained FNNs are practical for real-time execution on low-power devices as well. We provide some results to emphasize on these matters.

In this paper, we present a low-power smart device, i.e., Neblina™ system-on-module [1], with extensive hardware variants and expansion modules targeting Internet of Things (IoT) applications. The firmware stack on the base hardware features a configurable methodology for real-time MPR using inertial sensors. This involves 1) Shock aware segmentation, 2) Time-domain histogram extraction, and 3) A single-hidden-layer FNN classifier. The framework provides flexibility for developers to customize it for different applications. We evaluate the achievable accuracy, real-time performance, memory-usage, and power consumption of Neblina, when utilizing the proposed MPR on the fitness activity dataset in [4] with 17 subjects and 33 activities (~2.5M data samples). Using one IMU node attached to the right forearm, which has

978-1-5386-4980-0/19/$31.00 ©2019 IEEE

been mapped to Neblina, an overall accuracy of 98.23% has been achieved based on a random train-test procedure. The MPR engine resulted in a maximum latency of ~15.8ms, ~7% usage of the internal flash memory, and ~3% usage of the available RAM on Cortex M4F. We have also evaluated the proposed MPR framework on the RecoFit dataset [8], which involves 94 subjects and one IMU node (~14M data samples).

The rest of this paper is organized as follows. Section II describes the Neblina platform, its hardware variants, and its configurable framework for MPR. Section III presents the experimental results, while Section IV concludes the paper.

II. PROPOSED DESIGN

A. The Neblina Platform

Neblina's base hardware consists of low-power sensors, an on-chip memory and a Bluetooth Low Energy (BLE) chip with Cortex M4F to handle processing in real-time. Certain expansion modules for Neblina have also been developed in-house as depicted in Fig. 1. They provide support for wireless charging, inductive sensing, SD card storage, low-power wide area networks, high-g acceleration captures up to 400g, tactile feedback and energy harvesting with piezo-electric material. The tactile/haptics response has latencies in the order of a few milliseconds for improved user adoption. The wireless charging support is crucial, when the device is required to be packaged and integrated inside sports or industrial equipment, e.g., a football, where charging through USB is not feasible.

The base hardware is equipped with a sensor fusion library that features: 1) A computationally efficient 9-axis orientation tracking algorithm using inertial and magnetic sensors [2], and 2) A novel configurable MPR framework using inertial sensors, which is presented throughout the rest of this section.

B. Configurable MPR Framework

The proposed MPR engine is shown in Fig. 2. It performs *segmentation*, *feature extraction* and *classification*. While these steps can be tackled in numerous ways [3, 8], we utilize a unique configurable method, which based on our practical experience has shown great promise for IMU-based MPR in sports and consumer wearable domains.

B.1. Shock-Aware Segmentation

The segmentation process is based on an overlapping sliding window over the 3-axis accelerometer and 3-axis gyroscope sensors, which have initially passed the factory calibration/filtering techniques [2]. For motion segments characterized by an impact, e.g., a tennis or hockey shot, we can optionally add a centered peak constraint to the segmentation configuration. This would require a global peak with high enough intensity at the center of the segment t_{center}:

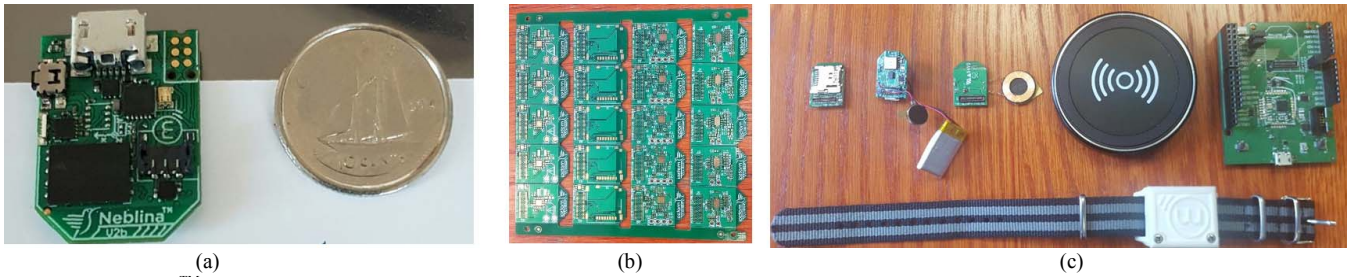
$$\max_{t\in Window}\big(F(t)\big) = F(t_{center}) \geq F_0, (F_0 > 0), \text{ or}$$
$$\min_{t\in Window}\big(F(t)\big) = F(t_{center}) \leq F_0, (F_0 < 0), \quad (1)$$

where F can refer to the overall acceleration magnitude or an individual axis and F_0 is the minimum peak intensity. The host device, e.g., PC or mobile phone, configures the sliding window length T and its incremental step size (See Fig. 2), as well as the optional peak configuration in Eq. (1), i.e., max/min mode, the peak function F and its minimum intensity F_0, by sending a single BLE command to Neblina. The setting will be stored in M4F's non-volatile internal flash memory.

It is notable that for MPR purposes, we discard the magnetometer and Euler angles (quaternion data) due to the following reasons: 1) Magnetometers are majorly affected by the magnetic disturbance within the environment, e.g., metal objects, mobile phones, monitors, etc. Hence, if we utilize these sensors to build classifiers, the models will not be cross-validated well against magnetic disturbance. 2) The pitch and roll angles are indirectly derived by a fusion of accelerometer and gyroscope data [2], and if the classifier can derive arbitrary nonlinearities from the input data stream, the immediate need to use these angles as independent inputs for MPR will be eliminated. An example for such classifiers is a neural network with at least one hidden layer, which can represent an arbitrary differentiable nonlinear function [5], and 3) The yaw angle and the true geographic direction of movements is often useless for analysis of transient motion patterns. Furthermore, variations in yaw over a short period of time can be tracked using gyroscopes and the gravity vector.

B.2. Feature Extraction

In the next step, we perform a time-domain histogram feature extraction over the selected IMU stream (See Fig. 2). We have chosen histogram features as opposed to the alternative plethora of statistical time-domain features as explored in [3, 8], due to the following two reasons: 1) A time-domain histogram, when created with high enough bins, will be an accurate representation of the distribution of the IMU data for a given segment and 2) Time-domain histogram extraction is low cost in terms of both latency and memory



(a) (b) (c)

Figure 1. Neblina™ base hardware and its expansion modules: (a) base hardware (22.2mm×16.5mm), (b) expansion base boards (from the left): 1) piezo driver and tactile feedback, 2) micro-SD card, 3) inductive sensing, 4) wireless charging, and (c) hardware variants (from the left): 1) SD card expansion module, 2) base hardware with a haptics device for tactile feedback, 3) wireless charging expansion module, 4) wireless charger, 5) development board.

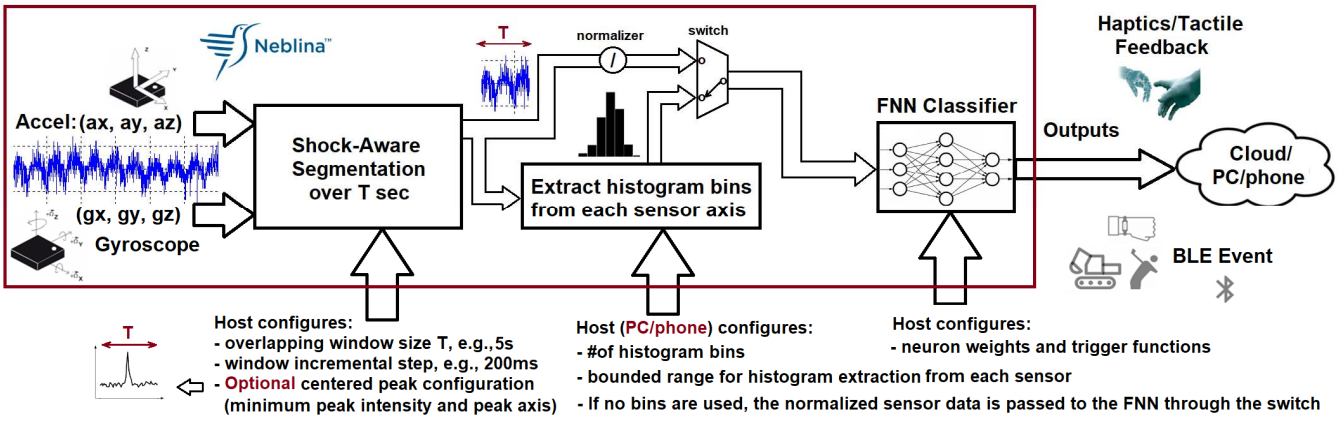


Figure 2. Neblina's configurable motion pattern recognition framework using IMU sensors. The host application on PC or mobile-phone configures: 1) a shock aware segmentation with an overlapping sliding window, 2) bounded sensor range with arbitrary bins for time-domain histogram feature extraction from IMU segments, and 3) neuron connection weights and trigger functions for a single-hidden-layer FNN classifier.

usage, which makes it a great fit for real-time execution on M4F. The number of histogram bins (histogram resolution) and the max/min sensor readings for histogram extraction (data range) are configurable by the host. Note that using a smaller sensor range, a higher bin resolution can be achieved. The histogram bins are normalized through a division by $(f \times T)$, where f is the sampling rate and T is the window length in seconds. If we utilize a centered peak option with a short window length T, we can optionally pass the sensor data directly to the classifier after a normalization, e.g., division by the maximum absolute sensor reading, as shown in Fig. 2.

B.3. Classification

Classification is performed using a single-hidden-layer FNN, whose weights and trigger functions are set by the host. The reason behind choosing such an FNN is twosome: 1) Its ability to derive arbitrary nonlinearities from its inputs [5], and 2) Efficiency in terms of both latency and memory usage for real-time execution on Cortex M4F, where the neuron weights are stored in the non-volatile internal flash memory and are treated as read-only values. The host can select among Sigmoid, Tanh, Relu, and Softmax trigger functions [5] for the hidden/output layer of the FNN. This configuration alongside neuron weights are sent to Neblina using BLE commands.

The FNN outputs in Fig. 2 can trigger user messages, e.g., tactile feedback on Neblina or audio/video feedback on the host using BLE notifications. Complementary niche heuristics can also be developed to for instance count the repetitions of a specific exercise. Namely, the predicted activity alongside the IMU stream can be passed to another engine to compute the repetitions using for instance auto-correlation [8]. Note that for the motion patterns characterized by an impact, e.g., a tennis shot, the use of the optional centered peak automatically delivers the repetition of each pattern, since only the segments with centered impacts are passed to the FNN.

III. Experimental Results

In this section we evaluate the MPR in Fig. 2 based on some existing human activity datasets. In our experiments, we use a 5s window sliding at 200ms (4.8s overlaps), alongside 20 histogram bins per sensor axis. This generates a total of 120 inputs for the FNN. The Sigmoid (Softmax) function is also used as the trigger function in the hidden (output) layer. The batch size of 32 has also been chosen for training the FNNs. The other configurations (number of hidden/output neurons, sensor range and epochs) are tuned based on the given dataset.

First, we consider the dataset from [4] with 17 subjects and 33 activities. We target the ideal IMU placement scenario [4]. Among the nine IMUs attached to different body joints, we have selected the device on the right forearm and mapped it to Neblina. The accelerometer (gyroscope) range for histogram extraction is set to $\pm 16g$ ($\pm 2000dps$). We also used 60 (34) hidden (output) neurons for the FNN. Due to the highly correlated 4.8s overlapping windows, we randomly chose only 10% of the dataset (~22K datapoints) for training the FNN. Using 1000 epochs, an accuracy of 99.9% is achieved.

FNN versus KNN: We validate the trained FNN using the test dataset (~207K datapoints) reaching an overall accuracy of 98.23%. To provide comparison with a KNN classifier, which is effective for activity recognition [3-4], we have chosen $k = 3$ with the same histogram features and train-test datapoints. The accuracy found by KNN is 97.66%. The per-activity F1 scores for both FNN and KNN classifiers are shown in Fig. 3. FNN slightly outperforms KNN. While these classifiers showed promise for this test, they resulted in similar low F1 scores below 75%, when targeting the leave-one-out cross-subject validation. These results can be improved by involving more subjects and (or) more IMU nodes. While FNN and KNN deliver similar results, for large datasets, KNN requires significant memory usage and suffers from long latency, which makes it infeasible for real-time execution on Cortex M4F. This is not the case for the FNN classifier.

Memory Usage and Latency: The segmentation, histogram extraction and FNN operation have been configured on Neblina for real-time execution. The memory usage in bytes and latency of these operations are reported in Table I. The neuron weights consume ~7.1% of the internal flash, while the overall RAM/stack usage for MPR is ~3% on Cortex M4F. The FNN operation with the latency of ~15.8ms is executed

once every 200ms for each sliding window, while the fast histogram computation is based on continuous update of bins per IMU sample, which is performed every 20ms at 50Hz sampling rate. Extracting histogram bins is fast, memory-efficient and powerful for MPR alongside a well-trained FNN.

Power Consumption: The static current consumption of Neblina, which uses a 5V supply voltage, has been measured as ~2.4mA, while streaming the FNN outputs at 5Hz over BLE. This corresponds to ~41 hours of battery operation considering a small 100mAh battery attached to Neblina.

Support for Larger FNNs: Currently, the largest supported FNN on Neblina can consume up to 33% of the internal flash, e.g., A) an FNN with 300 inputs (50 histogram bins per axis), 130 hidden neurons, and 34 outputs, which consumes ~1% of RAM and delivers a 26ms latency, or B) an FNN with 480 inputs (80 bins per axis), 85 hidden neurons, and 34 outputs.

Histogram versus Other Features: We have also compared the framework in Fig. 2 with RecoFit [8], which involves a dataset with 94 subjects. We consider "Study B" with 4 classes. While RecoFit extracts 224 diverse features [8], we use 20 histogram bins per axis over $\pm 2g$ and $\pm 512dps$. The leave-one-out cross-subject validation is then performed using an FNN with 20 hidden neurons, which is trained for only 20 epochs per left-out subject. This process results in an average accuracy of 98.2%, which is comparable to "98.6%" reported in [8].

Finally, we target a custom set with 7 classes using all the ~14M datapoints in [8]. The activities are biceps curl, triceps extension, running, elliptical machine, jumping jacks and kettlebell swing, alongside a noise class representing all other activities/non-activities. We used the same ranges, 60 hidden neurons and 7 epochs, which often reaches a ~99.6% accuracy during training. The leave-one-out cross-subject validation was performed for all 94 subjects, and it delivered F1 scores between 83% and 90% for the activities as shown in Table II.

IV. Conclusion and Future Work

This paper presented a configurable low-power system-on-module, i.e., Neblina™, for IoT applications with extensive hardware variants and expansion modules. The firmware

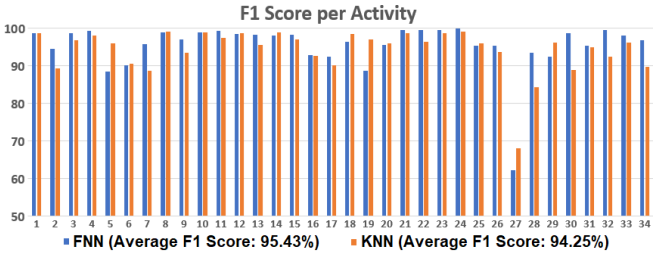


Figure 3. F1 Score per activity (33 activities plus one noise class#1) for the dataset in [4] under a 10-90 train-test procedure using two classifiers.

TABLE I. Memory Usage and Latency of Running the Proposed MPR Framework on Neblina Based on the Dataset in [4]

Method	Latency	Internal Flash Usage	RAM Usage
Segmentation and Histogram Extraction (*Proposed MPR*)	~30us	-	1623 (*2.5%*)
FNN Classifier (*Proposed MPR*) with (120-60-34) neurons	15.8ms	37336 (*7.1%*)	376 (*0.5%*)
KNN Classifier (*Conventional*)	-	*Out of Memory*	

TABLE II. Cross-Subject Validation F1 Scores per Activity Using the Proposed MPR Framework and the Dataset in [8]

Noise	Curl	Triceps	Run	Elliptical	JumpJack	Kettlebell
99	90.6	86.7	87	87.5	83.3	85.2

library provides a configurable real-time MPR framework based on only one IMU device. Experimental results explore how this framework can effectively be used for real-time classification of human fitness activities on Neblina based on some existing datasets. The use of one IMU node for activity recognition is significantly better than using multiple IMU nodes in terms of hardware cost, user adoption, etc. However, certain activities necessitate the use of multiple IMU nodes attached to different body joints. For that purpose, we plan on providing support for real-time MPR using multiple Neblina boards in a BLE mesh network. Furthermore, we aim to extend our platform support for 1) visual feedback, 2) biopotential sensors, and 3) advanced human computer interaction technologies, such as mid-air haptics [13].

Acknowledgement

We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) through a strategic partnership grant. We also thank Dr. Jean-Samuel Chenard and Hosein Nourani for their constructive comments and suggestions.

References

[1] Motsai page: www.motsai.com/products/neblina.

[2] O. Sarbishei, "On the accuracy improvement of low-power orientation filters using IMU and MARG sensor arrays", IEEE ISCAS 2016, pp. 1542-1545.

[3] M. Janidarmian, A. Roshan Fekr, K. Radecka, Z. Zilic, "A Comprehensive Analysis on Wearable Acceleration Sensors in Human Activity Recognition", Sensors (Basel). 2017; 17(3): 529.

[4] O. Banos, M. Damas, H. Pomares, I. Rojas, M. A. Toth, O. Amft, "A benchmark dataset to evaluate sensor displacement in activity recognition", Ubiquitous Computing, Sept. 2012, pp. 1026-1035.

[5] Bishop, Christopher M. (2006), "Pattern Recognition and Machine Learning", Springer.

[6] E. Velloso, A. Bulling, H. Gellersen, W. Ugulino, H. Fuks, "Qualitative Activity Recognition of Weight Lifting Exercises", ACM SIGCHI (Augmented Human), 2013, pp. 116-123.

[7] A. Eskandarinia, H. Nazarpour, M. Teimouri, M. Z. Ahmadi, "Comparison of Neural Network and K-Nearest Neighbor Methods in Daily Flow Forecasting", Journal of Applied Sciences, 10 (11), pp. 1006-1010, 2010.

[8] D. Morris, S. Saponas, A. Guillory, I. Kelner, "RecoFit: Using a Wearable Sensor to Find, Recognize, and Count Repetitive Exercises", ACM SIGCHI, Aug. 2014, pp. 3225-3234.

[9] S. Mehrang, J. Pietila, I. Korhonen, "An Activity Recognition Framework Deploying Random Forest Classifier and a Single Optical Heart Rate Monitoring and Triaxial Accelerometer Wrist-Band", MDPI Sensors, 18 (2), 613, Feb. 2018.

[10] U. Jensen, M. Schmidt, M. Hennig, et al. "An IMU-based Mobile System for Golf Putt Analysis", Sports Eng., June 2015, 18 (2).

[11] M. Ahmad, Waqarraza, Z. Omer, M. Asif, "A Participatory System to Sense the Road Conditions", Journal of Engineering and Manufacturing, 2017 (3), pp. 31-40.

[12] T. Roumen, et. al., "A comparative study of notification channels for wearable interactive rings", CHI 2015, pp. 2497–2500, 2015. ACM.

[13] St. Andrews, "UltraHaptics: multi-point mid-air haptic feedback for touch surfaces", ACM symposium on User interface software and technology, pp. 505-514, Oct. 2013.