CRNTC+: A smartphone-based sensor processing framework for prototyping personal healthcare applications

G. Spina†, F. Roberts†, J. Weppner††, P. Lukowicz††, O. Amft†
†*ACTLab, Signal Processing Systems, TU Eindhoven, The Netherlands*
††*Embedded Intelligence, DFKI/University of Kaiserslautern, Germany*
{*g.spina, amft*}*@tue.nl, f.j.m.roberts@student.tue.nl, jens.weppner@dfki.uni-kl.de, paul.lukowicz@dfki.de*

***Abstract*—While smartphone apps for health monitoring and patient support are of great interest to care providers and patients alike, suitable development and evaluation frameworks are currently lacking. We present and evaluate an Android open-source smartphone framework CRNTC+ for sensors data acquisition, signal processing, pattern analysis, interaction and feedback, based on the Context Recognition Network Toolbox (CRNT). CRNTC+ extends the original CRNT by providing components to read smartphone and external sensor data, supporting annotations, and various output components. Here, we formally evaluate CRNTC+ regarding extensibility, scalability, and energy consumption. We present study results where CRNTC+ was deployed in an application to detect epileptic seizures. Results showed that CRNTC+ is well-suited for prototyping health applications in real-life, where online sensor data recording and recognition is needed.**

I. Introduction

When utilising the various internal sensors and interconnection to external devices, modern smartphones can become on-body hubs for sensor data acquisition, processing, and feedback in personal health applications. The potential of smartphones has been widely recognised for medical training, monitoring, and assistance [1].

Evaluating smartphone-based solutions with patients often requires to develop applications and re-implement functionality. Smartphone-based software frameworks could reduce this implementation burden and enable developers to quickly prototype solutions. However, many existing smartphone frameworks lack essential features, including algorithms for sensor pattern recognition, signal processing, or software interfaces with different external sensors (see related work in the next section). Thus patients cannot choose and interoperate sensors. Since frameworks such as the Context Recognition Network Toolbox (CRNT) have been widely used with PC-based computer architectures [2], an integrating approach could leverage from existing algorithm implementations on smartphones.

In this paper, we present an open-source Android-based sensing and processing framework that integrates multiple sensing modalities especially suitable for patient monitoring. Our extended CRNTC+ framework integrates the complete CRNT functionality and provides additional input/output components to utilise smartphone-internal sensors and services as well as external devices attachable via wireless protocols, e.g. Bluetooth, ANT. The smartphone-specific framework and the CRNT were partitioned through dedicated interfaces and thus can be extended independently. Nevertheless, our partitioned design does not affect framework users during configuration and use. In particular, the paper makes the following contributions:

1) We introduce our CRNTC+ framework design and present its application-independent implementation.
2) We formally evaluate our framework and consider extensibility, scalability, and energy consumption.
3) In an exemplary prototype design, we evaluate CRNTC+ for detecting epileptic seizure events.

We chose epilepsy to evaluate CRNTC+, since patients suffering from epileptic seizures face various difficulties in daily life. In particular, major seizures may render patients unconscious and thus in potentially threatening situations. Thus, seizure detection could support patients and caregivers by alarming when a patient needs external help. Most sensor-based seizure monitoring approaches used single modalities focusing on limb acceleration or heart activity during seizures [3], [4]. Our results show that CRNTC+ can be used as a flexible solution for recording and detecting epileptic seizures during daytime using smartphones.

II. Related work

Several frameworks have been proposed to facilitate smartphone application prototyping. A number of smartphone data processing frameworks addressed specific applications, e.g. Pocket-Sphinx [5]. Pocket-Sphinx is a continuous speech recognition tool ported to Android. Other approaches include the Dandelion framework that uses RPC for message passing [6]. More recently, some general purpose sensor processing frameworks have been proposed. While a full review is beyond the scope of this paper, some examples are highlighted. The FUNF framework [7] and the SENSE Observation System platform[1], are able to acquire data over third-party sensors, supporting Bluetooth and ANT protocol transmissions. However, data analysis is primarily done

[1]The SENSE Observation System, http://www.sense-os.nl/home

through cloud processing and native processing algorithms have to be implemented with a proprietary API.

The Open Service Architecture for Sensors (OSAS) framework[2], is an event-based programming system for sensor networks. It facilitates sensor nodes programming in a sensor network. To implement solutions, functionality needs to be implemented using regular coding. Another software framework designed for rapid prototyping of activity recognition applications is CRNT [2]. CRNT uses a component-oriented architecture, where complete data processing chains can be configured by instantiating, parameterising, and interlinking components. Users of CRNT can thus develop an application without in-depth knowledge in programming. CRNT has been ported to many different PC-based platforms. The potential for utilising this framework in a smartphone environment has not been investigated so far.

III. Processing framework approach

Our CRNTC+ architecture design follows a component-oriented approach, including *Readers*, *Writers*, *Filters*, *Classifiers*, and others. The parameterisable components are incorporated through a run-time engine that can flexibly handle communication links between components in order to customise functionality.

The architecture is partitioned into smartphone-specific and generic data processing layers to separate platform API-dependent components and those for general data handling. Besides smartphone-specific components, including many *Readers* and *Writers*, CRNTC+ incorporates all generic data processing components of CRNT [2] for signal handling and pattern processing. Moreover, several CRNT *Writers*, such as for file logging and WLAN communication are directly usable. Figure 1 illustrates the layered design in a functional example. The basic architectural principles of component instantiation and data handling established for CRNT have been retained in the CRNTC+ framework. Component communication links can be routed within a layer and between layers. The architecture can be expanded by adding further components to both smartphone-specific and generic data processing layers. All components and communication links between them are configured and parameterised jointly through a JSON-based description. Hence, to design an application, components just need to be selected, parameterised, and interlinked only.

IV. Implementation

In our framework design and implementation we targeted extensibility, and scalability through convenient interfaces to add and customise components. Moreover, design efficiency is key to minimise energy consumption. Here, we detail

[2]OSAS, http://www.win.tue.nl/san/wsp/index.html

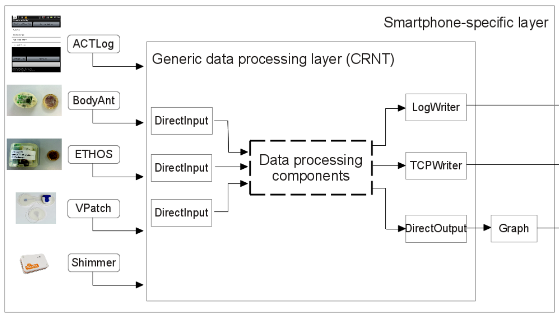

Figure 1. Functional overview of the CRNTC+ framework. *Reader* components are used to capture sensor data and user input , while *Writer* components serve to output information, e.g. through a Graph component or via a WLAN link. Our approach considers smartphone-specific (platform-dependent) and generic data processing layers.

the implementation of key component classes: *Readers* and *Writers*. Moreover, general implementation considerations for CRNTC+ on the Android platform are described.

Readers: To interface with sensors and devices via different communication standards *Readers* are used. Through *Readers*, various smartphone-integrated sensors can be recorded. Examples for external device interfaces include *BluetoothReader* and *ANTReader* components. Readers connect to devices specified in the component configuration. Depending on the sensor device protocol, data streaming is subsequently started and readings are decoded for further processing in the framework. E.g., *ANTReader* uses the ANT+ protocol to connect to sports or custom devices, such as BodyANT, ETHOS, and Vpatch. The *BluetoothReader* can be used, e.g., to interface to a heart rate belt or to Bluetooth accelerometers. Due to the phone APIs, *Readers* reside in the smartphone-specific layer of CRNTC+.

Writers: *Writers* encode data streams from CRNTC+ for further analysis and feedback. E.g., the *Graph* component provides a timeseries view on the phone's screen for reviewing sensor or feature data. *Writers* reside in smartphone-specific and generic data processing layers. E.g., data file logging to an SD card can be performed through the *LogWriter* component using generic POSIX calls, whereas the *Graph* component requires platform-dependent features.

Component instantiation: In CRNTC+ a JSON-based configuration is used to describe component instances, their parameters, and communication links. A JSON configuration is instantiated at runtime by matching a class definition of the component. The GSON library was used to convert JSON representation into Java Objects using string mapping. When a JSON configuration file is loaded, the type field representing the module is checked and, if it matches to a map key, this module is instantiated by using the class definition that is coupled to the key.

Reader for user annotations: To enable smartphone users annotating sensor data, CRNTC+ integrates an *ACT-Log* component, which works as a reader for user input.

ACTLog provides a configurable UI within CRNTC+ to capture annotations in pre-configured categories. To annotate data, the phone user needs to tap and hold a category label and then select a sub-category label instance from a configured list displayed. Annotations can be directly processed in CRNTC+ or stored to a labels file for subsequent analysis. *ACTLog* resides in the smartphone-specific layer.

***Between-layer communication**:* Besides direct within-layer communication, *DirectInput* and *DirectOutput* components are used as internal gateways to transfer data between framework layers. This design is needed to bridge between the different implementations of both layers: while the smartphone-specific layer uses native code of the Android platform, the generic processing is integrated as a library in the CRNTC+ application. A direct data communication between the layers is essential to minimise overhead and processing load compared to other communication forms between layers, such as RPC or TCP/UDP.

V. Framework characterization

To evaluate the CRNTC+ framework performance, we assessed extensibility, scalability, and energy consumption.

***Extensibility**:* We evaluated the ease of adding a new component and measured the steps necessary to create new *Readers* and *Writers*. Table I summarizes the extensibility evaluation results. Four steps were needed to add a new sensor *Reader* component and *Writer*, with 18 code lines and 22 code lines, respectively. Adding a new UI element requires five steps and 34 code lines.

Table I
EXTENSIBILITY ASSESSING CRNTC+ FOR ADDING COMPONENTS. ESTIMATIONS INDICATE THE SMALLEST EFFORT. FOR FUNCTIONAL COMPONENTS, ACTUAL EFFORT CAN BE LARGER.

Add Readers components

Step	Minimum lines of code	Other complexities
1. Extend Module class	3	Further subclassing
2. Extend ReaderClass class	13	Depending on sensor
3. Add Module class definition	1	None
4. Add ReaderClass class definition	1	None

Add Writers components

Step	Minimum lines of code	Other complexities
1. Extend OutputModule class	7	None
2. Extend OutputClass class	13	None
3. Add Module class definition	1	None
4. Add OutputClass class definition	1	None

Add user interface (UI) components

Step	Minimum lines of code	Other complexities
1. Extend GUIModule class	12	None
2. Extend MyTabActivity class	13	Retrieve GUIModule
3. Create icons	0	None
4. Create layout xml file	3	Depends on GUI structure
5. Create drawable xml file	6	None

While the actual complexity of adding components depends on the required functionality, the evaluation indicates the basic framework-specific requirements for an extension. It can be observed that UI elements requires the largest effort, since an icon is needed and the Android framework requires to handle life cycles of "Activities". Overall, the result indicates that the framework does not imply complex steps for functionality extension.

***Scalability**:* We evaluated scalability by incrementally adding, recording, and visualizing calibrated accelerometer data from Shimmer sensors. To assess performance we measured CPU usage and measurement jitter. Up to three sensors could be simultaneously recorded without loosing samples at a sampling rate of 200 Hz. When using four sensors, responsiveness of the UI reduced and CPU time for updating the UI decreased. This result suggests that three sensors could be safely recorded without loosing samples.

***Energy consumption**:* For the Epilepsy case study described in Sec. VI, two applications have been created for gathering sensors data and for seizure event detection. During the execution of both applications, energy consumption of the smartphone was monitored. With the full sensor configuration, battery level discharged by 80% during $\sim$6 hours of sensor recording. This result can be explained by the continuous data writing onto the SD card, decoding of data sent via Bluetooth, and continuous screen use for annotating data. It can be expected that reducing sensors will reduce energy needs. Similarly, online processing without storing to the SD card could increase battery life.

VI. Experimental evaluation

A. *Epilepsy evaluation study*

We evaluated the CRNTC+ framework in a case study to investigate data acquisition from multi-modal on-body sensors and recognising seizures. Since epileptic seizures often occur only sporadically in patients during daytime, two expert actors were asked to simulate five different seizures types (myoclonic, tonic, tonic-clonic, clonic, myoclonic tonic-clonic) during ten everyday activities, including lying in bed, getting dressed, scratching, drinking from a glass, brushing teeth, sit-ups, shaking hands, using mouse, typing on a keyboard, folding towels.

Heart rate data were acquired using a Shimmer ECG module [3], placed at the left upper arm. The module featured a 3D accelerometer too. Disposable electrodes were connected to the ECG module and attached to the participant's chest. Respiratory data were acquired using a Braebon [4], strap, placed around participants' thorax and connected to a second Shimmer ECG module. Another 3D accelerometer was placed at the right upper arm. Two full inertial motion units (one ETHOS and one Shimmer 9DOF) were placed at participants' left and right wrists. Data was acquired at two different sampling rates: for the Shimmer units 100 Hz, and for the ETHOS sensor 128 Hz. All the sensing modules were hold in place using adjustable velcro straps. Data from all sensors was recorded via Bluetooth (Shimmer) and ANT (ETHOS) using a Sony Ericsson Xperia active

[3]Shimmer, http://www.shimmer-research.com
[4]Braebon Respiratory Effort Sensor, http://www.braebon.com/

smartphone. A study observer was carrying the phone and used ACTLog to annotate the activities during recordings.

Since expert actors were recorded instead of epilepsy patients, heart rate and breathing pattern reacted with a delay and caused by the physical activity related to simulated seizures, rather than an actual seizure. Thus, ECG and respiratory data was considered to assess the CRNTC+ framework scalability, but excluded from further analysis.

B. *Epilepsy study results*

Approximately 40 min of continuous recording were acquired per participant. Data was segmented using a window size of 100 sa. Variance of the 3D accelerometers unit placed on both the upper arms and on the dominant wrist were analysed and used in a two-class classification (seizure against non-seizure). All analyses were performed using the CRNTC+ and a frame-based evaluation. First, to train an offline kNN classifier (k=3), 500 samples per class were randomly selected from data from both actors. Remaining samples were used for testing. Subsequently, to test the feasibility of real-time seizure detection, the configuration was tested online. For practical application we considered that the system should alarm within one second from the start of a seizure. To satisfy the real-time constraint of the online evaluation, the training set needed reduction to 100 samples and only one sensor was used. The performance limiting factors for the real-time analysis were the Shimmer sensor data transmission and the classifier processing.

The offline detection test using 3 accelerometers, showed a class specific accuracy of 74% for seizure event and 64% for non-seizure. For the one-sensor configurations, 72% and 59% was obtained for the upper left arm, 76% and 63% for the upper right arm, and 81% and 62% at the wrist, for seizure events and non-seizure times respectively. Subsequently, the right wrist sensor was chosen to test online recognition performances. Figure 2 summarizes the performance results. For the online recognition, the reduced training set resulted in a deteriorated performance, with 78% for seizure events and 55% for non-seizure times. After revising the training set to the core seizure phase with high motion intensity only, performance improved to 86% for seizure events and 78% for non-seizure times.

VII. Conclusion and Further Work

We proposed and evaluated a new framework for smartphone-based sensor data recording and processing, which emphasises extensibility and leverages the widely used CRN toolbox for generic data processing algorithms. The new CRNTC+ was implemented in a layered framework design. Our formal evaluation and study results showed that CRNTC+ is versatile to handle various multi-modal sensors and recognition solutions, which are essential to prototype patient care solutions with phones. While the present investigation focused on assessing feasibility of CRNTC+ for epilepsy detection in daily life, further work is needed to optimize the detection by evaluating additional algorithms and evaluation in larger studies.

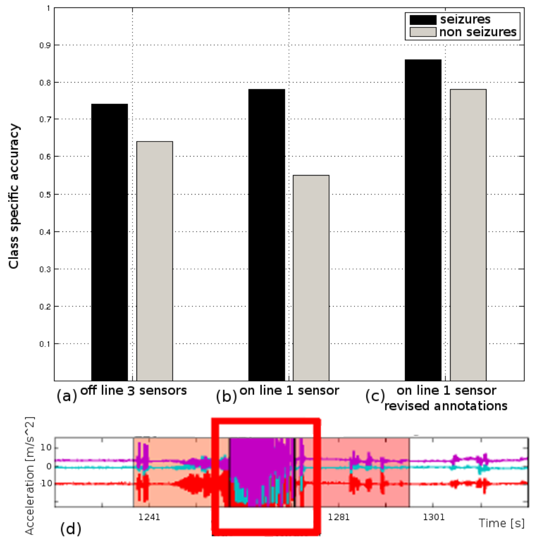

Figure 2. Epilepsy seizure detection performance using the CRNTC+ framework. (a): offline, using three 3D acc. units placed on the upper arms and dominant wrist. (b): online, using one 3D acc. unit at wrist. (c): online, using one 3D acc. unit at wrist with revised training data (see main text for details). (d): Annotation example for a Tonic-clonic seizure. The red square marking indicates the seizure part used as training data for the results in panel (c).

Acknowledgement

The authors are thankful to the expert actors participating in the Epilepsy evaluation study. This work was supported by the EU FP7 Marie Curie Network iCareNet under grant number 264738 and the iCare4COPD project of Agentschap NL under contract number PNE101005.

References

[1] O. I. Franko and T. F. Tirrell, “Smartphone app use among medical providers in acgme training programs.” *J Med Syst*, 2012.

[2] D. Bannach, P. Lukowicz, and O. Amft, “Rapid prototyping of activity recognition applications,” *Pervasive Computing, IEEE*, 2008.

[3] J. Lockman, R. S. Fisher, and D. M. Olson, “Detection of seizure-like movements using a wrist accelerometer,” *Epilepsy and Behavior*, 2011.

[4] S. Nasehi and H. Pourghassem, “Real-time seizure detection based on eeg and ecg fused features using gabor functions,” in *ICBMI*, 2011.

[5] D. Huggins-Daines, M. Kumar, A. Chan, A. Black, M. Ravishankar, and A. Rudnicky, “Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices,” in *ICASSP*, 2006.

[6] F. X. Lin, A. Rahmati, and L. Zhong, “Dandelion: a framework for transparently programming phone-centered wireless body sensor applications for health,” in *Wireless Health*, 2010.

[7] N. Aharony, W. Pan, C. Ip, I. Khayal, and A. Pentland, “Social fmri: Investigating and shaping social mechanisms in the real world,” *Pervasive Mob. Comput.*, 2011.