An educational and research kit for activity and context recognition from on-body sensors

Daniel Roggen, Marc Bächlin, Johannes Schumm
Thomas Holleckz, Clemens Lombriser, Gerhard Tröster
Wearable Computing Laboratory, ETH Zürich
Email: daniel.roggen@ife.ee.ethz.ch

Abstract—We present an educational and research kit to support hands-on teaching and experience of real-time activity or gesture recognition from on-body sensors. The kit is comprised of: wireless wearable sensor nodes for motion and ECG sensing; software infrastructure for synchronized data acquisition from multiple sensors, data visualization, signal alignment, and synchronized signal/video exploration; algorithm demonstration and education software with a hidden Markov model-based activity recognition system and a low-latency gesture recognition for a platform game; support hardware for the annotation of user activities from a wireless keypad, and for prototyping of other wireless sensor nodes. All hardware and software is open-source.

I. INTRODUCTION

Activity recognition [1] is a key to many mobile, wearable and pervasive computing applications, such as gestural interfaces [2] or industrial worker’s assistance [3], or wellness [4].

Two aspects motivate our work:

- The recognition of some simple activities (e.g. locomotion, postures) is reaching a maturity level where it can enter the realm of industrial and consumer applications.
- The recognition of complex hierarchical activities and manipulative gestures remains a research challenge.

In the first case, activity recognition should become more mainstream to enable scenario-driven applications. In the second case undergraduate and graduate students must understand the current challenges and limitations of activity recognition. In both cases, we believe that achieving a “first hand” experience with an actual activity recognition system is the key. Our objective is to develop a set of tools that supports eduction in activity recognition methods for e.g. undergraduate students, and other persons not necessarily in the core field of activity recognition (e.g. in industry, gaming, bio-medical fields). This allows through personal interaction to find out about the existing challenges and thus to more quickly identify the relevant research questions, such as developing new algorithms, new sensors, new context representations or new applications.

A. Contribution

We present an educational and research kit to support hands-on teaching and experience of human activity recognition (online, real-time) from on-body sensors. The focus lies on the methods enabling activity recognition. We do not attempt to provide a system optimized for best recognition performance. Rather our goal is a system that allows to understand the relations between the variations of input parameters in the recognition system and the effects on the resulting performance of the system. The key design considerations are to achieve openness and ease of use, to facilitate focusing on the methods for activity recognition. All hardware and software is open-source so that tools can be reused and improved.

II. RELATED WORK

There are other efforts to produce “toolkits” for activity and context recognition. The Georgia Tech Gesture Toolkit [5] consists of a series of interconnected command line scripts. BeTelGeuse is an open source platform to collect data from internal phone sensors and Bluetooth sensors [6]. SPINE (Signal Processing in Node Environment) is a framework for the design of mobile-phone based activity-aware applications using Zigbee sensor nodes [7]. TITAN is a service-graph approach to the dynamic distribution of context recognition application on sensor nodes [8]. The CRN Toolbox is an open-source data-flow oriented acquisition and processing framework [9] based on a processing block interconnection representation.

Many sensor nodes suitable for activity recognition are commercially available. Xsens technologies (www.xsens.com) sells wired inertial sensors often used for human activity sensing. Crossbow sells sensor nodes with support for inclusion in a business logic. Their wireless node TelosB has been used in wearable computing with custom sensor extensions for human motion sensing. SparkFun Electronics (www.sparkfun.com) sells hardware components for robotics. Some motion sensing systems can also be used on-body. Many physiological sensors, in particular ECG (or ExG) and respiration sensors are available from companies such as Polar (www.polar.fi), CamNtech (www.camnitech.com), g.tec (www.gtec.at), or TMS International (www.tmsi.com). These devices usually lack in openness, in the form of available schematics or PCB layouts (e.g. physiological systems listed above), or possibility to reprogram the nodes (e.g. XSens, Sparkfun sensors). The CodeBlue project that develops wearable physiological sensing is one exception following an open hardware approach.
III. DESIGN DECISIONS

Our design decisions are motivated by: openness, ease of use, integration of the hardware/software, and emphasis on educational aspects to allow a quick start to the thematic of activity recognition. The key novelty lies in the interoperability of the hardware and software elements for educational purposes while still allowing many software and hardware elements to be used for other ends. Another key aspect is that the system allows to vary a wide range of parameters of the activity recognition chain, and to immediately test the result through hands-on experimentation in a single integrated environment.

The Atmel AVR microcontrollers is the core of the nodes as there exist an open-source C/C++ compiler, high-level libraries, operating systems, and cheap programming tools.1 To foster ease of use the sensor nodes must interface with a large range of devices and software. We selected Bluetooth wireless communication as it is available on most laptops and mobile devices.2 We use the Bluetooth Serial Port Profile (SPP, based on the RFCOMM protocol) to connect from the host to the sensor nodes. Thus the sensor node appears a virtual serial port. This has several advantages:

- SPP provides error correction and retransmission, thus enhancing communication reliability without increasing host-side software complexity.
- SPP is supported in all major operating systems (Linux, Windows, Windows mobile, Maemo, OSX).
- The sensor appears as a standard serial COM port. Virtually all languages and OS allow to access them.
- Many languages provides direct Bluetooth APIs with additional libraries: Java with JSR82, Python with PyBlueZ, .NET, or C/C++ with BlueZ under Linux.
- Scientific development environment are commonly used for signal processing and machine learning to prototype and test activity recognition systems. Matlab (and Simulink), can directly open the serial port as a file, or use the serial object part of the Instrument Control toolbox to connect to the sensors. Octave, the open-source equivalent of Matlab, provides similar capabilities.

Finally, the kit must be integrated and streamlined. At the sensor node level, this means that all nodes share the same architecture and the same firmware. Thus, there is only the need to understand how one node functions to get a clear picture on all the device’s operations. All nodes share the same communication protocol, yet the protocol is flexible and allows various number of channels with different bit-resolution. The protocol can be easily implement in many languages (see section V where it is implemented in C++, Java, and Matlab). A protocol description format is used to configure all the softwares of the kit. Thus, new nodes can be developed and the complete software suite can be reused without additional programming. Finally, a purpose build station allows easy charging and programming of all nodes.

At the software level care was taken so that sensors can be used besides this kit, by providing data acquisition drivers for use with other systems. Also, several of the infrastructure software are designed to be generic and usable with other sensors.

IV. HARDWARE COMPONENTS

The kit contains: NTMotion:AccGyro, acceleration and rate of turn sensor node; NTECG: ECG sensor; NTKBD, annotation keypad; NTProto, prototyping node; and a charging/programming station.

Figure 1 illustrates the overall hardware architecture, including the interface for charging and programming. The common characteristics are the following:

- ATMEAL ATmega324P micro-controller
- AmberWireless Bluetooth Module (BlueNiceCom III or BlueNiceCom IV), connected to the primary AVR UART.
- 2 LEDs (PWR/ERR and IND)
- MAX1595 voltage regulator
- Rechargeable 300mAh Li-ion battery (larger is possible)
- Battery lifetime of more than 6h with a 300mAh battery
- Board-to-board connector to the charging/programming station. It also allows communication with the PC over the secondary AVR UART.
- Rapid-prototyping plastic packaging for all nodes

\[\text{Charging and Programming Station} \quad \text{Wireless Node}\]

1The AVG-GCC compiler. The open-source library AVR-LIBC with a rich subset of the C standard library and AVR-specific functions. Operating systems such as FreeRTOS (open source preemptive multitasking OS) and TinyOS (open source cooperative multitasking OS often used in wireless sensor networks). Programming software such as WinAVR (from Atmel) or AVRDUDE (open-source). Programming interfaces selling for less than 100$, such as the Atmel’s STK500, or similar products from companies such as e.g. Olimex (www.olimex.com).

2802.15.4 or proprietary protocols are generally preferred in WSN research. However they require additional hardware, they are not available on mainstream devices, and they distract from the focus on the algorithmic aspects of activity recognition.

A. Motion sensor - NTMotion:AccGyro

NTMotion:AccGyro is a sensor node with a 3-axis ADXL330 accelerometer and a 2-axis IDG650 gyroscope (fig 2). This combination is commonly used for physical context acquisition (manipulative gestures, modes of locomotion, physical activity, gait analysis). It extends a previous acceleration-only sensor [10]. The motion sensor is
Sensor data are sampled at regular intervals. At each sample, node is indicated by LED codes: connection and disconnection, charging slot and the PC (e.g. for debugging). A wired acceleration-only sensor is also available for rapid deployment in a classroom (not depicted here, 43x36mm). It minimizes the overhead of a battery-based system, which requires a charging station, and wireless communication, which requires discovery and pairing, in favor of a USB connection which works out of the box with all major operating systems. It is exchangeable with the wireless motion sensor as they follow the same protocol [10].

B. ECG sensor and optional respiration sensor - NTECG

This sensor streams the ECG measured between two electrodes (fig. 3). The signal is picked-up by a low gain (10x) instrumentation amplifier (INA326). After low-pass filtering ($f_c=35Hz$) the signal is amplified amplified (380x, LMP2014) for reading by the AVR ADC converter. A hardware 8th-order elliptic filter (MAX7407, $f_c=35Hz$) may be activated. A connector allows to connect a 2-pin resistive stretch sensor to measure breathing, in a voltage divider configuration. This node can be used for applications requiring custom on-body signal processing, such as heart-rate variability analysis or artefact detection. It builds upon previous analog circuitry developed for non-mobile use [11].

C. Annotation device - NTKBD

The annotation device, in the form of a wireless keypad, is used to mark events of interest during recordings, such as specific activities (fig. 4). It behaves as a virtual “sensor” by streaming key presses to the wearable computer.

D. Prototyping node - NTPROTO

This node allows to prototype custom circuit while re-using the node architecture of the kit (fig. 5). It is derived from the NTKBD device, but excludes the keyboard. It offers 7 user configurable analog and digital pins. The custom circuit can be placed on top of the sensor node PCB and connected to the 7-pin expansion connector.

E. Charging station

A charging/programming station provides one charging and programming slot, and 5 pure charging slots (fig. 6). The charging circuit is a MAX1555. LEDs indicate the charging state. A USB connector with an FTDI FT232R UART-USB transceiver allows to communicate between the node in the charging slot and the PC (e.g. for debugging).

F. Firmware

All sensors share the same firmware. The state of the sensor node is indicated by LED codes: connection and disconnections with a host, data streaming (connected and streaming), standby (not connected), or errors.

All nodes send data with a frame-based streaming format. Sensor data are sampled at regular intervals. At each sample, a frame is created and sent to the host. The frame includes: header, data channels, and an optional checksum. The header provides a means to synchronize the frame decoding, useful if data is lost due to communication errors. The checksum ensures the integrity of the frame. Since each sensor module contain different sensor kinds, the exact format (number of channels, channel bit-width) is specific to each sensor. The largest frame is that of the motion and rate of turn sensors when all channels are transmitted, with 27 bytes. Other configurations lead to smaller frames.

All the software tools support this streaming format in a generic manner by means of a format definition string of the format HDR;<cH0><cH1>...[/cksum]. This string indicates what is the frame header, the number of data channels and the data channel size, and the optional presence of a checksum.

For instance, the format definition string DX3;cc-s-sbl0b12;x indicates that the frame header is DX3, that there are 6 data channels, and that a 8-bit LRC checksum is used (x). A 16-bit Fletcher checksum is indicated by X or the checksum can be left out altogether. The channel sizes are indicated by the letters in cc-s-sbl0b12. c, s, i, bnn represent 8-bit, 16-bit, 32-bit, or nn-bit little endian channels. The prefix minus indicates a signed channel.

All sensor module support a menu-based configuration. The configuration mode is accessed over USB within 2 seconds of the power-up or over Bluetooth within two seconds of the connection. All sensors and the keypad support the following configuration options: sample rate (8Hz to 256Hz), packet counter format (8 or 16 bit), checksum type, and streaming of test data (rather than the sensor data).

V. SOFTWARE

A. DScope: digital oscilloscope

DScope is a cross-platform Qt-based digital oscilloscope with configurable number of scopes, number of traces per scopes, individual trace colors, individual scope horizontal and vertical zoom, and refresh rate. It can connect directly to the sensors of the kit, and to TCP/IP sources. Besides the streaming format described above, it supports a plain text format too (one line per sample, space separated numbers for each channel). Acquired signal data can be exported to a file.

B. CRNT Toolbox driver

A generic FBReader driver allows to acquire sensor data from the CRN toolbox (see section II). FBReader attempts to reconnect to the sensor if a disconnection occurs (e.g. when the receiver moves too far away from the sensors) to minimize data losses during long recordings. FBReader is generically configured with a format string, com port, and reconnection delay. Thus it is suitable for existing or new sensor nodes. The CRN toolbox is available for PC but has also been used on iphone, Nokia internet tablets, and other mobile devices.

3This sensor has 3 raw acceleration channels, 3 calibrated acceleration channels, 2 rate of turn channels, 2 amplified (higher sensitivity) rate of turn channels, and a zero-rate channel.
C. SensHub: Data recorder, server and labeler

SensHub can acquire data from multiple sensors (fig. 8). It can combine the data from multiple sensors in a single stream, while ensuring that the data of all sensors remains temporally aligned despite data loss or communication bursts that are common in wireless links. It provides offline merge, which yields the highest temporal alignment quality. For real-time applications it also provides online merge with different algorithms, including a rate-control mechanism similar to [12]. In this mode there is a trade-off between merge latency and signal alignment quality (with 100ms to 200ms good alignment has been observed with up to 7 sensors). SensHub can store sensor data in individual files and in a single merged file. It also acts as a TCP server for individual and merged sensor data for access from other software. Acquired and streamed data can be labeled with the keyboard. SensHub is a cross-platform Qt application running on Windows, Linux and OSX, and with adaptation on Maemo-based systems (e.g. Nokia N900).

D. Java BluetoothGateway

The Java BluetoothGateway is a tool to discover the sensors, record their data into a file, or act as a TCP server for a remote client. It receives data from multiple sensors and synchronizes and resamples their data streams. Upon lost connection, BluetoothGateway attempts to reconnect to the sensor. It runs on a Windows Mobile device or a PC.

E. SynScopeV: Signal-video alignment, exploration and documentation

Sensor data may be recorded by multiple computers (e.g. a wearable computer for on-body sensors and a desktop for ambient sensors) and with different sample rate, or start time of the recording. When sensors are recorded in individual files, it is important to align the signals between them before applying activity recognition algorithms.

SynScopeV’s main purpose is the interactive alignment of signals and the alignment of signals with video footages (fig. 9). Any number of signals or videos sources are supported. Alignment is achieved by estimating the offset and resample rate between sources, as defined by reference points provided by the user in the graphical interface. The signals and video can then be visually explored and all the sources scroll according to the link relations. SynScopeV can resample signals to a common sample rate to merge aligned signals, and it offers several options to export relations between signals and videos for use in other tools.

SynScopeV can generate videos to document datasets by putting side by side synchronized video footages and sensor signals. See http://vimeo.com/8704668 for an example.

VI. EDUCATIONAL AND DEMONSTRATION SOFTWARE

Two softwares are provided for educational and demonstration purposes. They follow a typical activity recognition chain. An activity recognition chain is a relatively standard sequence of processing steps, that has emerged as the dominant approach in activity recognition from on-body sensors

Fig. 2. Packaged motion sensor.
Fig. 3. Wireless ECG sensor
Fig. 4. Annotation keypad
Fig. 5. Prototyping sensor node.
Fig. 6. The charging/programming station with 6 charging slots, one of which allowing node programing and data.
Fig. 7. DScope with user-configurable scopes presenting sensor communication over USB.
Fig. 8. SensHub with the specification of sensors to acquire (left), the real-time channel merge parameters (top), and the link status (bottom).
Fig. 9. SynScopeV, with 3 videos (top) and 2 sensor signal scopes (bottom).
This sequence is: data acquisition from the sensors; segmentation of the data stream into sections of interest likely to contain an activity; feature extraction within these sections to reduce their dimensionality; classification of the features into a set of output classes (activities); and optional “null-class” rejection (see fig. 10). The methods are not codified and various algorithms may be used at each stage of the chain. For educational value, two softwares demonstrate different methods. Hidden Markov models (HMM) are frequently used for hierarchical activity recognition. The first software allows to teach the design choices and trade-offs in using HMMs in an activity recognition chain with explicit segmentation. The second software is designed to be more visually appealing, by combining activity recognition with a game engine. It allows to teach choices and trade-offs to design a low-latency activity recognition chain (important for HCI or gaming scenarios), with a simple NCC classifier operating on a sliding window, and with multiple sensors. Both softwares are designed for use in a classroom or laboratory session, and the second can be used as a demonstrator to a more general audience.

A. Real-time activity recognition using hidden Markov models

A Matlab-based software allows to use separate discrete hidden Markov models to classify activities using the acceleration values of the motion sensor. The software acquires directly the sensor data, visualizes data in real-time, and does real-time recognition of activities and visualization of recognition likelihoods. In the classroom we use it to demonstrate gesture recognition, with the sensors typically placed on the forearm at the wrist, but it can also be used for other type of movements.

First, the user records training data. Many parameters are available for training HMMs modeling the training gestures. The parameters include among others the discretization of the acceleration values, number of states, architecture of the HMM, number of iterations in the Baum-Welch training process. After training the software executes the recognition chain. Two segmentation approaches are provided. In energy-based segmentation, a segment starts when the hand starts to move, and ends when it stops moving (identified by the energy in the acceleration signal). In rest-position segmentation, a segment starts when the arm is not within the tolerance of a trained rest position, and ends when the arm returns to that position. At the end of a segment, the Viterbi algorithm is used to find the likelihood for the gesture to be modeled by each HMM. The HMM yielding the highest likelihood indicates the gesture that has been observed.

The main educational aspects are: i) the trade-offs between the complexity of the HMM (number of states, observations, architecture), the computational time and the need for sufficient training data, ii) the trade-offs between sensitivity and specificity of the segmentation method and the resulting types of classification errors, iii) the sensitivity to sensor orientation and placement variability, and gesture execution variability.

B. Low-latency real-time gesture recognition in games

A C++-based software allows to recognize simple HCI gestures to control the linux open source computer game SuperTux (http://supertux.lethargik.org/), using one motion sensor typically placed on the forearm (or two, on the left and right forearms).

First, training gestures are recorded for each of the controls of the game (e.g. move left/right, jump). When several sensors are used the software automatically detects which sensor is relevant for the training gesture.

The recognition chain consists of energy-based segmentation (see above). Two feature extraction approaches offer different latency-accuracy trade-offs. In the first approach, features are computed on the acceleration signal on a window aligned to the start of the segment, with a length specified by the user (typically about 100ms long). This assumes that gestures are different in their starting phase, and allows to achieve lower-latency than the second approach. In the second approach, features are computed on the entire segment (i.e. from start to end of gesture). Therefore classification can only be performed when the gesture is completed, which translates into higher latency and potentially higher accuracy. The features that are computed include among others mean, variance, energy distribution, zero-crossing or mean-crossing rate of the sensor signal. The sensor signal can be the acceleration along a specific axis, the acceleration magnitude, or the rate of turn. The user selects by means of a configuration file which features are used. Eventually, the features are classified with a nearest class centroid (NCC) classifier, which is a simple, low-complexity classifier well suited for embedded implementation. To avoid matching outliers a “null-class rejection” rejects the gesture if the features are located further away from the class centroid than a threshold distance. Eventually the detected gesture is sent to the modified SuperTux game as a UDP packet, thus essentially simulating a key press.

The main educational aspects are: i) introduction to the topic of human-computer interaction; ii) obtaining low-latency gesture classification; iii) understanding what are meaningful features to classify the desired gestures.

VII. Conclusion

We presented an educational and research kit for the recognition of activities from on-body sensor. This kit focuses on educational and research aspects related to the activity recognition chain (i.e. the methods and algorithms allowing activity...
recognition). Consequently the underlying design choices emphasize ease of access to sensor data, ease of deployment on multiple platforms without additional hardware, simple and robust hardware, and an integrated hardware and software solution. The key novelty is that a wide range of parameters of the activity recognition chain can be varied and immediately tested through hands-on experimentation. Thus students can learn the design trade-offs and the relations between the input parameters of the recognition chain, and the resulting characteristics of the recognition system. In our experience these design choices allow to focus on teaching activity recognition methods within 2x2 hours exercise sessions with up to 20 students during the Wearable Systems lecture at ETH Zürich.

Besides this, the hardware and software has been used in a wide range of research projects, thus showing its versatility: in the EU project OPPORTUNITY we acquired data from 24 on-body and object integrated sensors to recognize complex activities in sensor rich environments [14]; we used the kit in a wearable assistant for Parkinson disease patients with the freezing of gait syndrome [15]; in a snowboarding assistant [16]; in benchmarking new activity recognition methods [17]; for sensor fusion [18]; and in gait-based biometrics [19]. These deployments confirmed the ease of use of the system with a charging station allowing easy node recharging with a simple slide-in solution, the high robustness of the system, and the convenience of using a standard wireless protocol available on a wide range of mobile platforms.

We made sure the kit components remains generic despite forming an integrated system. This eases the development of additional sensor nodes. By conforming to the generic communication protocol new sensors can be seamlessly used with the software suite. Many software (e.g. SynScopeV, DSocpe) are general purpose and can be used with other data sources or sensors than this kit. The sensor nodes can also be used in other data acquisition systems.

The software, hardware and packaging described in this article is available as open hardware and open software for reuse or modification with credits given to the original work (this article). Table I indicates where the components are hosted.

<table>
<thead>
<tr>
<th>Component</th>
<th>Main repository</th>
<th>Availability</th>
</tr>
</thead>
</table>

TABLE I
AVAILABILITY OF THE EDUCATIONAL AND RESEARCH KIT

ACKNOWLEDGMENT
Part of this project has been financed by the Swiss-funded project Educati-