Social Signal Processing for Dummies

Ionut Damian, Michael Dietz, Frank Gaibler, Elisabeth André
Human Centered Multimedia
Augsburg University, Augsburg, Germany
{lastname}@hcm-lab.de

ABSTRACT

We introduce SSJ Creator, a modern Android GUI enabling users to design and execute social signal processing pipelines using nothing but their smartphones and without writing a single line of code. It is based on a modular Java-based social signal processing framework (SSJ), which is able to perform realtime multimodal behaviour analysis on Android devices using both device internal and external sensors.

CCS Concepts

•**Human-centered computing** → *Ubiquitous and mobile computing systems and tools;*

Keywords

social signal processing; behaviour analysis; realtime; online

1. INTRODUCTION

Recent advancements in computing have enabled the development of wearable computers and sensors. Such devices allow us to take computational platforms away from enclosed spaces and perform in-the-wild data processing. One particular area which can benefit from this shift is social signal processing. Considering the computational power and sensing capabilities of modern smartphones, it is possible to envision systems which perform state-of-the-art social signal processing on the go.

Unfortunately, the majority of (freely available) social signal processing solutions still rely on powerful personal computers to do the processing. While there are some exceptions, none of them offer a flexible and accessible way for creating powerful multimodal social signal processing pipelines. Platforms such as BeTelGeuse [4] and Dynamix[1] lack support for advanced signal classification techniques and also require programming expertise to operate. On the other hand, MobileSSI [3], the Android-compatible UNIX port of the popular OpenSSI framework [5], struggles with basic input/output tasks, such as interfacing with bluetooth sensors,

ICMI'16, November 12–16, 2016, Tokyo, Japan
ACM. 978-1-4503-4556-9
http://dx.doi.org/10.1145/2993148.2998527

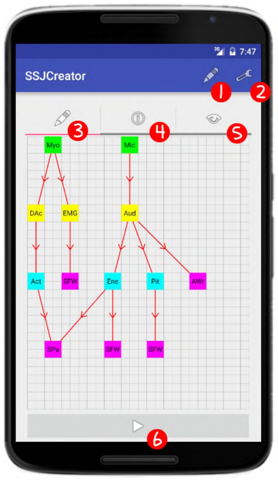

Figure 1: SSJ Creator — An Android app for building and executing social signal processing pipelines.

due to the incomplete C++ API of Android.

To address these issues, we implemented the SSJ framework, a Java-native solution for social signal processing fully compatible with modern mobile devices. Moreover, in an effort to put social signal processing at the finger tips of as many researchers, students or aficionados as possible, we developed a user-friendly Android app for building and executing SSJ pipelines (see Figure 1). For the first time, users are able to design and perform social signal processing experiments on a smartphone and without writing a single line of code.

2. THE SSJ FRAMEWORK

This section introduces SSJ, a framework for extracting, processing and classifying multimodal social signals with the help of external sensing hardware. SSJ has been inspired by Wagner's Social Signal Interpretation (SSI) Framework [5] for windows, and thus borrows several design principles from it. However, unlike SSI, SSJ is written natively in Java, giving it access to the entire Android API. Furthermore, thanks to the abstraction layer offered by Android's Java virtual machine, SSJ is device independent, and thus able to run on virtually all devices in the Android ecosystem. This allows an unprecedented flexibility and mobility for doing social signal processing. By combining this with a robust multithreaded architecture and aggressive memory management guidelines, SSJ is able to also run smoothly on older and less powerful hardware. So far, SSJ has been successfully deployed on common smartphones (Android API 16 - 24), smart glasses (e.g. Google Glass, Lumus DK-40) and smart watches (e.g. Moto 360).

At the heart of SSJ lies the concept of pipelines for handling and processing data. In a pipeline, data flows from one side to the other during which it sequentially passes various processing steps which transform the data in realtime. Every pipeline starts with a *Sensor* and a *SensorProvider*. A *Sensor* is a component which manages the connection with an internal (e.g. accelerometer, GPS) or bluetooth-connected sensor device (e.g. Myo, Microsoft Band). Each sensor device output is managed by a *SensorProvider*. The *SensorProvider* receives the data from the sensor device and forwards it into the pipeline. Once inside the pipeline, the data is passed through each individual registered processing step, or *Transformer*, in a sequential order. A *Transformer* transforms one or more inputs in an effort to filter or extract characteristic features from the raw data. The processed data is then pushed back to the pipeline. The end of the pipeline is usually marked by a *Consumer*, which, unlike a *Transformer*, does not have an output, only one or more inputs. Most commonly, *Consumers* represent the culmination of the pipeline, carrying out the actual classification of the data using either simple threshold or more advanced machine learning-based approaches. However, *Consumers* can also be used to communicate the behaviour analysis results with the user using graphs or camera painters, or with other applications (e.g. a feedback generator in a social augmentation scenario [2]) using Bluetooth, network or SSJ's own event interface for communication between applications on the same device. At the current time, there are more than 20 different sensors and over 30 transformers and consumers implemented in SSJ. Furthermore, thanks to an active community, additional components get added regularly.

3. SSJ CREATOR

In order to allow users to build and execute SSJ pipelines on the go from the palm of their hands, we implemented an Android app called *SSJ Creator*. The app is built around the classic build–test–iterate concept, allowing users to quickly build and test pipelines.

To facilitate this, SSJ Creator uses a proven tab-based layout. The first tab features a prominent pipeline view where the current pipeline configuration including the components, the connection between the components and the data flow is illustrated (Figure 1 ③). Users are able to edit pipelines by adding new components using the edit menu (Figure 1 ①), altering the options of each components by tapping on their icon in the pipeline view or dragging and dropping components to create links between them. Moreover, SSJ Creator also supports saving and loading of pipelines using the file menu (Figure 1 ②). Pipelines are stored using a human-readable XML format akin to SSI's XML pipelines [5]. Once the user finishes building a pipeline, she or he can launch it using the play button (Figure 1 ⑥).

When the pipeline is running, the second tab provides a view of SSJ's log (Figure 1 ④). This facilitates easy inspection and efficient debugging of SSJ pipelines. Based on the configuration of the pipeline, new tabs will be dynamically created and placed to the right of the log tab. More precisely, every output component (e.g. signal or camera painters) will receive a dedicated tab (Figure 1 ⑤) which will allow the user to inspect the data output.

4. SIMPLE PIPELINE EXAMPLE

Say that, for a study we need to record and analyse the users' movement so that we can later compare between user groups. This can be achieved with an SSJ Pipeline running on a smartphone which the users carry in their pockets. To create the pipeline, we first add an *AndroidSensor* and, from the options menu (accessed by tapping on the sensor), we configure the sensor to output the linear acceleration. We then add an *AndroidSensorProvider* and connect it to the sensor by dragging the sensor over the provider. To process the raw acceleration data, we add the *OverallActivation* transformer and connect it to the provider. This computes the amount of movement per frame (the frame size can be changed from the options menu). Finally, we add a *SimpleFileWriter* consumer to store the data on the SD card as well as a *SignalPainter* to visualize the data in the app, and connect both to the transformer. The pipeline is now ready for testing.

This simple pipeline can be further extended by including additional sensors (e.g. a Microsoft Band for heart rate) or a *SocketWriter* to stream the data to an external application. Furthermore, if we build a model with the recorded data (e.g. using SSI's [5] Naïve Bayes classifier), we can then use SSJ to automatically classify the activity of the user in realtime.

5. CONCLUSION

SSJ and SSJ Creator provide the first complete solution to designing, building and executing social signal processing pipelines on mobile devices using an easy-to-use graphical user interface. Both SSJ and SSJ Creator are open source and freely available for download[1].

Acknowledgments. The work was partially funded by the German Federal Ministry of Education and Research (BMBF) - project "EMPAT" (FKZ 16SV7229K).

6. REFERENCES

[1] D. Carlson and A. Schrader. Dynamix: An open plug-and-play context framework for android. In *Proceedings IOT*, pages 151–158, 2012.

[2] I. Damian, C. S. Tan, T. Baur, J. Schöning, K. Luyten, and E. André. Augmenting social interactions: Realtime behavioural feedback using social signal processing techniques. In *Proceedings CHI*, pages 565–574, 2015.

[3] S. Flutura, J. Wagner, F. Lingenfelser, A. Seiderer, and E. André. MobileSSI: Asynchronous fusion for social signal interpretation in the wild. In *Proceedings ICMI*. ACM, 2016 (in press).

[4] J. Kukkonen, E. Lagerspetz, P. Nurmi, and M. Andersson. BeTelGeuse: A platform for gathering and processing situational data. *IEEE Pervasive Computing*, 8(2):49–56, 2009.

[5] J. Wagner, F. Lingenfelser, T. Baur, I. Damian, F. Kistler, and E. André. The social signal interpretation (SSI) framework - multimodal signal processing and recognition in real-time. In *Proceedings of ACM MULTIMEDIA*, 2013.

[1]http://http://hcmlab.github.io/ssj/