OPEN ACCESS
OPUS AUGSBURG

Universität Augsburg
Universitätsbibliothek

The SSJ framework: augmenting social interactions using mobile signal processing and live feedback

Ionut Damian, Michael Dietz, Elisabeth André

Angaben zur Veröffentlichung / Publication details:

Damian, Ionut, Michael Dietz, and Elisabeth André. 2018. "The SSJ framework: augmenting social interactions using mobile signal processing and live feedback." *Frontiers in ICT* 5: 13. https://doi.org/10.3389/fict.2018.00013.

Nutzungsbedingungen / Terms of use: **CC BY 4.0**

Dieses Dokument wird unter folgenden Bedingungen zur Verfügung gestellt: / This document is made available under these conditions:
CC-BY 4.0: Creative Commons: Namensnennung
Weitere Informationen finden Sie unter: / For more information see:
https://creativecommons.org/licenses/by/4.0/deed.de

frontiers in ICT

CODE
published: 20 June 2018
doi: 10.3389/fict.2018.00013

The SSJ Framework: Augmenting Social Interactions Using Mobile Signal Processing and Live Feedback

Ionut Damian, Michael Dietz and Elisabeth André*

Human Centered Multimedia, University of Augsburg, Augsburg, Germany

Thanks to recent technological advancements, powerful computers and versatile sensor arrays can now be integrated in small wearable devices. This paper introduces the Android framework SSJ, which makes use of such devices to augment social interaction. To this end, sensor data is first used to analyze the user's behavior in realtime. Based on the analysis, SSJ then sends live multimodal feedback to the user while they participate in a social interaction. With the help of advanced feedback strategies, SSJ can minimize the disruption of the social interaction while still helping the user improve the quality of their behavior.

Keywords: social signal processing, mobile signal processing, live feedback, behavior analysis, code:java, code:android

OPEN ACCESS

Edited by:
Carlos Duarte,
Universidade de Lisboa, Portugal

Reviewed by:
Susanna Spinsante,
Università Politecnica delle Marche, Italy
José Baptista Coelho,
Universidade de Lisboa, Portugal

****Correspondence:***
Michael Dietz
dietz@hcm-lab.de

Specialty section:
This article was submitted to Human-Media Interaction, a section of the journal Frontiers in ICT

Received: *08 February 2018*
Accepted: *04 June 2018*
Published: *20 June 2018*

Citation:
Damian I, Dietz M and André E (2018) The SSJ Framework: Augmenting Social Interactions Using Mobile Signal Processing and Live Feedback. Front. ICT 5:13. doi: 10.3389/fict.2018.00013

1. INTRODUCTION

Social behavior is a crucial element of our daily lives. We constantly rely on it to navigate various situations, e.g., going shopping, communicating with friends and family and collaborating with colleagues. While most of the time we take it for granted, in some cases our ability to exhibit and regulate our social behavior can reach its limits. For example when holding a presentation, emotions and stress can drastically impact the quality of our social behavior. Moreover, persons suffering from certain difficulties can struggle to perform basic social behavior such as maintaining vocal loudness (Ramig et al., 2001) or correctly articulating their voice (Kanner, 1968). In these cases, social augmentation approaches (Damian et al., 2015, 2016) can help users improve their social behavior while participating in the social interaction.

This paper introduces the SSJ software framework for creating social augmentation systems. At the core of the social augmentation concept lies the behavioral feedback loop. It allows the system to continuously analyze the behavior of the user, and then provide the user with feedback on the quality of their own behavior as well as how to improve it. To achieve this, SSJ supports state-of-the-art mobile social signal processing and live feedback techniques. More specifically, SSJ enables the recording, processing and classification of social signals in realtime on mobile devices. For this, it is able to interface with various device internal as well as external (Bluetooth-connected) sensors. SSJ is also capable of delivering live multimodal feedback to the user with the help of various output devices such as head-mounted displays, headphones or smart armbands.

SSJ is published under the GNU General Public License v3 and freely available for download[1],[2]. It is based on the Social Signal Interpretation framework for Windows (Wagner et al., 2013). Yet SSJ has been completely redesigned with the challenges of social augmentation in mind. To this end, SSJ has been developed from the ground up for the Android ecosystem. Thus, it is able to run

[1]https://hcm-lab.de/ssj/
[2]https://doi.org/10.5281/zenodo.1242843

on virtually all Android devices (API 16 or newer) including smartphones, tablets, smart glasses (e.g., Google Glass, Lumus DK-40, Epson Moverio) and smart watches (e.g., Moto 360, Samsung Gear), allowing an unprecedented flexibility and mobility for performing social signal processing. The first version of SSJ has been released to the public in March 2016. Since then, 15 new releases spanning over 800 commits followed. SSJ is packaged as a single Android *.aar* library that can be easily integrated in Android applications. Additionally, the framework can also be operated using the *SSJ Creator* Android application[1]. It allows persons without a technical background to work with SSJ using a modern graphical user interface. More precisely, it enables the design and execution of signal processing pipelines and social augmentation systems without writing a single line of code. The application has been added to the Google Play Store in October 2016. Since then, it has been downloaded more than 200 times by users from over 10 countries. Out of all downloads, over 50 installations are still active. This high conversion rate and the broadness of the install base suggests that the application is useful for the general public as well, despite its roots as a research instrument.

2. AUGMENTING SOCIAL INTERACTIONS

The overall incentive for augmenting social interactions is to help users participating in social interactions achieve a more advantageous outcome. For example, social augmentation could aid users deliver better and more compelling speeches by assisting them with their voice modulation or gesture usage. Job seekers with social dysfunctions could use social augmentation to land a better impression during job interviews. A social augmentation system could also be used to help persons who suffer from various disabilities, such as autism or Parkinson's disease better cope with social situations. Starting from these use cases, two concrete goals can be formulated.

1. *Generate awareness of one's own body.* As argued in the first section, it is often the case that one's perception of their own behavior is not in line with how it is perceived by others. An increased awareness of one's own behavior would facilitate the detection and recognition of such unwanted behaviors, and thus represents the first step toward correcting them.
2. *Improve the quality of one's own behavior.* Besides generating awareness, the social augmentation should also guide the user toward a behavioral state which benefits them more in the social interaction. For instance, during a job interview, the social augmentation should attempt to change the behavior of the user with the aim of maximizing their chance at employment. To achieve this, the information delivered by the augmentation to the user should be sufficient to elicit a behavioral change. Yet, the augmentation must also be mindful of the limited and fragile nature of human attention and how too much information (or badly delivered information) can negatively impact the overall quality of the user's behavior.

2.1. Behavioral Feedback Loops

To accomplish these goals, behavioral feedback loops have been chosen as the driver behind social augmentation. In the simplest of terms, a feedback loop occurs when the output of a system is repeatedly and continuously fed back to the system as input, thus forming a closed loop. From the point of view of social augmentation, feedback loops are particularly interesting due to their self-regulating nature. Thus, the goal of generating self-awareness presented above can be directly translated to a feedback loop structure. The user's behavior (output) is recorded, processed and fed back to the user (input) continuously, generating awareness of one's own behavior. Now, through intelligent and goal-oriented manipulation of the feedback loop, the behavior of the user can be steered toward a more beneficial state for the social interaction.

The implementation of the behavioral feedback loop in SSJ is illustrated in **Figure 1A**: The user behavior is first analyzed in realtime and then, based on its quality, feedback is automatically generated and delivered to the user.

2.2. Adapting to User and Context

Humans are complex beings operating in a complex world. A visual message might be correctly perceived and decoded by one

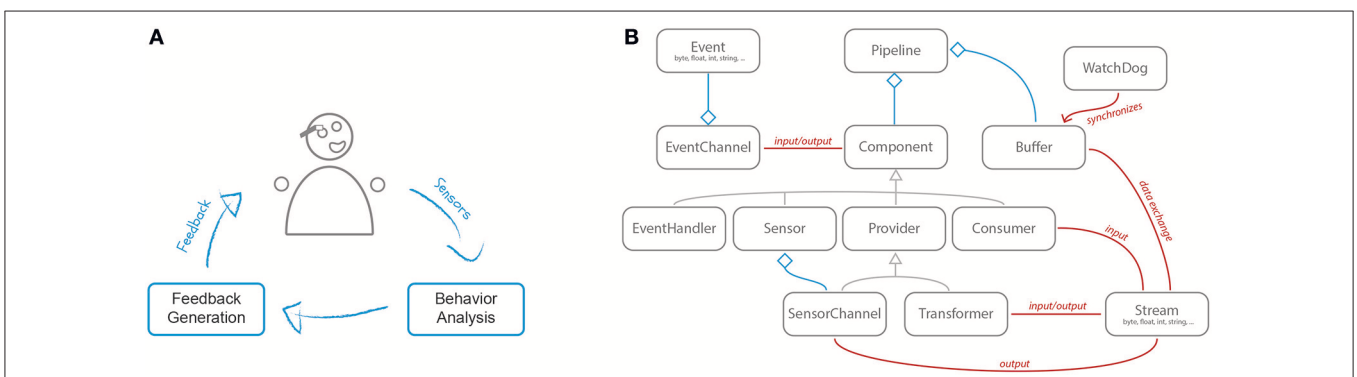

FIGURE 1 | Basic concepts: **(A)** the behavioral feedback loop featuring its two main components: behavior analysis and feedback generation; **(B)** component overview of the SSJ framework.

person but completely ignored by another one, simply because for one of the persons the stimulus resembled the poached egg she had for breakfast on the first day of her honeymoon. A beep might be flawlessly perceived at the beginning of a social interaction but missed 10 min later because of an increase in background noise. Thus, the feedback (as well as the whole behavioral feedback loop) needs to be able to adapt to the user, the context and the scenario. This is similar to the problem described by Arroyo et al. (2002). While studying how users respond to different types of stimuli, they found that "more notable than the differences between modalities was the differences between people" and that "subjects' sensitiveness depended on their previous life exposure to modalities."

To tackle this issue, SSJ supports three adaptation methods. First, based on user activity, the social augmentation can be automatically turned on or off. Second, once the augmentation is active, realtime adaptation techniques can be used to continuously adjust the feedback to the user's behavior. Finally, low level mechanisms manage the timing of the individual feedback events to make sure they are delivered at opportune moments.

3. ARCHITECTURE

SSJ is a highly modular framework and is composed out of various components (see **Figure 1B**). To create an application (e.g., for social augmentation), multiple components are strung together to form a pipeline. There are four main component types: sensors, transformers, consumers and event handlers. A sensor is responsible for pushing data into the pipeline. Most commonly this data is directly extracted from a physical sensor device. A single sensor can be the source of multiple signals and thus can be associated with multiple sensor channels. For example, a camera provides both audio and video data.

Whereas sensors have multiple outputs, consumers have one or more inputs. This allows them to receive data from multiple components and process it simultaneously. For instance, a classifier can use both audio and video data to perform multimodal classification. Transformers have both inputs and outputs. Thus, they receive data, transform it, and then push the transformed data back into the pipeline. A typical example of a transformer is filters. They process the data stream by filtering out unwanted artifacts (e.g., noise). Another example is feature extractors, which refine the data to extract meaningful characteristics (e.g., pitch extracted from a raw audio signal). The fourth component type, the event handler, is unlike the others as it specializes in sending and receiving discrete events. Whereas all other components require continuous connections with a fixed sample rate between each other, events are spontaneous and are not bound to any sample rate.

A pipeline is a chain of components where the outputs of some components are matched to the inputs of other components. Thus, pipelines represent directed acyclic graphs where data flows along the edges and is processed in the nodes. Pipelines can also fork and fuse and two pipeline branches may run completely in parallel. To facilitate correct processing across multiple channels, SSJ maintains all branches of a pipeline synchronized at all times.

4. EXAMPLE: PROVIDING FEEDBACK IN RESPONSE TO VOICE QUALITY

To demonstrate the ability of SSJ to execute complex behavioral feedback loops, this section will present a tutorial on how SSJ can be used to create a system which provides live feedback in response to the quality of the user's behavior. Let's assume we are about to hold a presentation in front of a large audience and are a bit unsure about the quality of our voice. Using SSJ, we can train a model to classify our speech quality, and then provide feedback whenever it detects an insufficient quality. For this, we only need an Android smartphone with SSJ Creator[3] installed. While SSJ is capable of interacting with various other sensing and feedback devices which enable the analysis of manifold social signals and the delivery of multimodal feedback, in an effort to make this tutorial accessible to every reader, we will solely rely on an Android smartphone. This tutorial is structured in four parts:

1. Build a pipeline to collect and annotate data from the user in two conditions: speaking normally and speaking very fast.
2. Train a model using the collected data.
3. Build a pipeline which uses the model to perform live classification.
4. Configure a feedback strategy to provide feedback in response to the classified speech quality.

4.1. Data Collection

To collect the necessary data for training a model, a data recorder is needed[4]. Using SSJ Creator, we can build a pipeline which extracts sensor data, processes it and then stores the data locally on the device (final pipeline shown in **Figure 2A**).

For this, in SSJ Creator, we first add a *Microphone* sensor and an *AudioChannel* using the + menu (see **Figure 2B**). We now connect the sensor to the channel by long pressing on the yellow sensor box and dragging and dropping it over the channel. Following this, we add an *Intensity* transformer, connect it to the *AudioChannel* and configure it (by tapping on it) to take in 1.0 s of audio data per frame and to compute the mean intensity over the entire window (see **Figure 2C**). We also add a *Pitch* transformer and configure it to take in 0.1 s of data from the audio channel. The output of the *Pitch* transformer is further processed using an *Average* transformer configured with a 1.0 s input window to match the output rate of the *Intensity* transformer. Finally, we connect the *Intensity* and *Average* transformers to a *FileWriter* consumer for storing the data on the SD card. The name of the output file can be configured in *FileWriter*'s options menu. Set it to "data" and leave the path at its default value (it will create a new timestamped folder in "/sdcard/SSJ/"). The final pipeline layout is shown in **Figure 2A**. Before starting the pipeline, we should also configure our annotation tab. This will

[3]https://hcm-lab.de/ssj/

[4]Alternatively, example data can be downloaded from https://doi.org/10.5281/zenodo.1249289

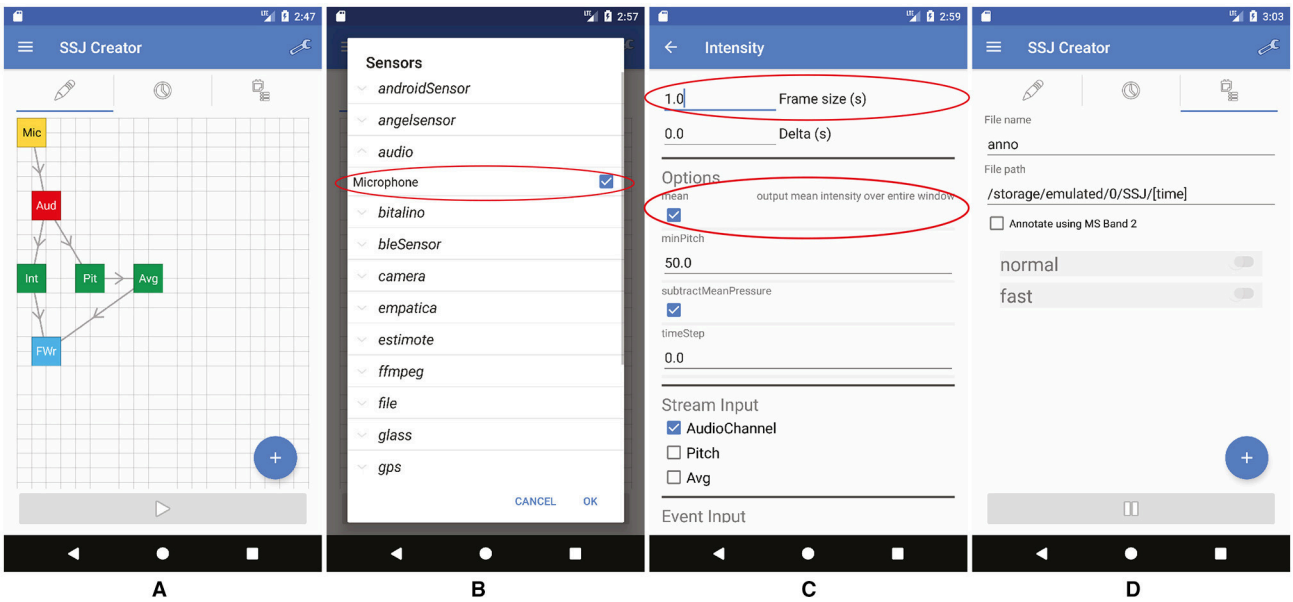

FIGURE 2 | Building a data recorder using SSJ Creator: **(A)** the final pipeline, **(B)** adding the sensor, **(C)** configuring the *Intensity* transformer **(D)** configuring the annotation tab.

allow us to annotate the start and end of an event (e.g., speech episodes) while recording. For this simply add two classes and name them as shown in **Figure 2D**. To start the pipeline, we simply press the "play" button. We should also save the pipeline for later use with the help of the menu in the upper right corner.

Once we have our data recorder in place, we need to collect data. For this simply start the pipeline, switch to the annotation tab and record some examples of normal speech and some examples of fast (unintelligible) speech. Make sure you also toggle the annotation buttons when you start and stop speaking.

4.2. Training a Model

Once we have collected our data, we can train a model for the automatic classification of speech quality. This can be achieved using the "Train Model" screen (**Figure 3A**) accessible from the main menu (the three horizontal lines at the top left corner).

First, add a new data source and configure it (by tapping on it) to point toward the recorded data ("data.stream") and the annotation ("anno.annotation"). If you conducted multiple recordings, you can add multiple data sources. To make the recognition more robust, we should check the "fill up empty spaces" checkbox to also provide the model with non-speech data (i.e., the data before, between and after your annotated utterances). This will allow it to discriminate speech from background noise. Next, one can select the model type one wishes to train[5]. To start the training, simply press the "Train Model" button.

[5]Currently, SSJ v0.7 only supports Naive Bayes models for training. Additional model types can be trained on a PC using SSI (Wagner et al., 2013)

4.3. Realtime Classification Using SSJ

The next step is to incorporate the trained model in our pipelines. For this, we load the recording pipeline and remove the *FileWriter* component (by dragging it over the trash symbol). We now add a *Classifier* consumer and a *NaiveBayes* model (**Figure 3B**). We also need to connect the *Intensity* and *Average* transformers to the *Classifier* and the *Classifier* to the *NaiveBayes* model. To use the model we trained in the previous step, we need to configure our *NaiveBayes* model to point toward it (select the *.trainer* file).

The pipeline is now complete and can be tested (don't forget to also save it for later use). Once the pipeline runs, you should now see the current classification output in the log tab. For this tutorial we opted to use Naive Bayes due to its simplicity, yet SSJ also supports SVM and Neural Network models for inference.

4.4. Live Feedback

The final step of the tutorial is to extend the live classification pipeline with a feedback component that is able to deliver feedback to the user in response to the classification. SSJ supports three feedback modalities: visual, auditory and tactile. For this tutorial, we will use visual feedback to avoid disrupting the audio signal. To achieve this we add a *VisualFeedback* event handler to the pipeline and connect the *Classifier* to it (**Figure 3C**). The feedback can be configured by using the options panel of the component (**Figure 3D**). Setting the option *eventNames* to "fast" will cause the feedback to only be triggered once the "fast" class yields the largest classification likelihood. Using the *feedbackIcon* option, one can select an image to be shown as feedback. The *duration* option controls how long the feedback is shown on the screen while the *fade* option controls the fading animation.

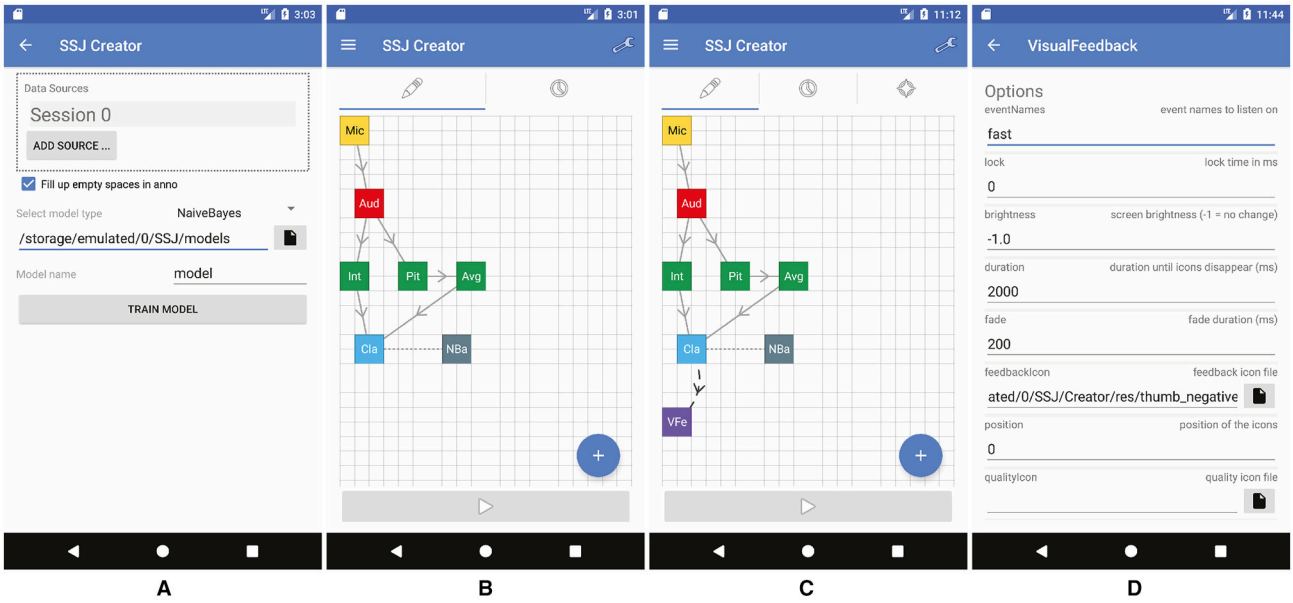

FIGURE 3 | Preparing the pipeline for feedback generation: **(A)** training a model using the recorded data, **(B)** replacing the *FileWriter* with a classifier and a model, **(C)** adding the feedback **(D)** configuring the feedback.

Once the pipeline is active, it will continuously extract audio data from the microphone, classify it into *normal*, *fast* and *GARBAGE* with the help of the pre-trained model, and display a visual feedback every time the audio is classified as *fast*.

Alternatively, one could split up the pipeline between the smartphone and an HMD (e.g., Google Glass). The main pipeline performs the classification on the computationally superior smartphone and sends the results over Bluetooth to the HMD. A second pipeline is executed on the HMD. It receives the Bluetooth events and displays the feedback. Moreover, more complex feedback strategies can be defined using the *FeedbackCollection* component. This allows one to define dynamic feedback behavior, which changes over time depending on how the user behaves. For example, if the user ignores the visual feedback, additional modalities (audio or tactile) can be employed to increase the prominence of the feedback.

5. LIMITATIONS

Although the computational power of smartphones has rapidly improved over the past decade and has reached a level where mobile social signal processing is feasible, certain tasks such as training neural networks or processing videos with high frame rates and resolutions are currently still limited by the processing capabilities of the underlying hardware. This also applies to the performance of the SSJ framework, which is mostly impacted by the processing power of the utilized smartphone and the selected sample rate of the components within the pipeline.

Another constricting factor of mobile devices is their lack of a continuous power source and their reliance on finite batteries. While SSJ applies Android's built-in energy saving techniques in an effort to minimize power consumption, its usage can still affect battery life. For example, running the classification pipeline presented earlier requires on average 0.348 W power, causes 22.17% CPU load and leads to a battery life of 8 h 52 min on a Google Nexus 6P smartphone[6]. Due to the application of energy and performance optimizations and the usage of platform-specific APIs and SDKs for communicating with internal and external sensors, the SSJ framework is currently also only available for the Android operating system.

6. CONCLUSION

This paper introduces the SSJ framework. SSJ allows the creation of mobile social signal processing pipelines and, through the use of live feedback components, social augmentation systems. Such systems record and process the behavior of the user and then provide feedback on its quality, effectively enabling *in-situ* social skill coaching. Thanks to its flexibility and reliance on off-the-shelves hardware, SSJ can also be used in other scenarios. For example, social scientists or curious users can quickly build a pipeline for collecting data of a particular event. The recorded data can then be inspected on a computer using compatible analysis tools, such as NOVA (Baur et al., 2013). Furthermore, due to its advanced communication capabilities, SSJ can be used to create

[6]Measured with Qualcomm Trepn Profiler https://developer.qualcomm.com/software/trepn-power-profiler

remote monitoring or warning systems. For instance, two smartphones can be used to create a baby monitor with an incorporated alerting function, which triggers whenever a loud noise is detected or the baby's heart rate exceeds a predefined threshold.

AUTHOR CONTRIBUTIONS

ID and MD are the main developers of the SSJ framework. The framework was created as part of ID's PhD (Damian, 2017). EA supervised the entire work as well as the drafting of the article.

FUNDING

The work was partially funded by the German Ministry for Education and Research (BMBF) within the projects Glassistant (FKZ 16SV7267K) and EMPAT (FKZ 16SV7229K).

ACKNOWLEDGMENTS

Part of this manuscript is based on ID's PhD thesis (Damian, 2017). It is the only medium that this content has appeared in. The inclusion in this publication is in line with the policies of the University of Augsburg.

REFERENCES

Arroyo, E., Selker, T., and Stouffs, A. (2002). "Interruptions as multimodal outputs: which are the less disruptive?" in *Multimodal Interfaces (ICMI), Conference Proceedings (IEEE)* (Pittsburgh, PA).

Baur, T., Damian, I., Lingenfelser, F., Wagner, J., and André, E. (2013). "Nova: Automated analysis of nonverbal signals in social interactions," in *Human Behavior Understanding, Workshop Proceedings, Vol. 8212 of Lecture Notes in Computer Science* (Barcelona: Springer).

Damian, I. (2017). *Social Augmentation Using Behavioural Feedback Loops*. Phd Thesis, Universität Augsburg, Augsburg, Germany.

Damian, I., Baur, T., and André, E. (2016). "Measuring the impact of behavioural feedback loops on social interactions," in *Multimodal Interaction (ICMI), Conference Proceedings*, (Tokyo: ACM), 201–208.

Damian, I., Tan, C. S., Baur, T., Schöning, J., Luyten, K., and André, E. (2015). "Augmenting social interactions: realtime behavioural feedback using social signal processing techniques," in *Human Factors in Computing Systems (CHI), Conference Proceedings* (Seoul: ACM), 565–574.

Kanner, L. (1968). Autistic disturbances of affective contact. *Acta Paedopsychiatr.* 35, 100–136.

Ramig, L., Sapir, S., Countryman, S., Pawlas, A., O'Brien, C., Hoehn, M., et al. (2001). Intensive voice treatment (LSVT) for patients with parkinson's disease: a 2 year follow up. *J. Neurol. Neurosurg. Psychiatry* 71, 493–498. doi: 10.1136/jnnp.71.4.493

Wagner, J., Lingenfelser, F., Baur, T., Damian, I., Kistler, F., and André, E. (2013). "The social signal interpretation (SSI) framework - multimodal signal processing and recognition in real-time," in *Multimedia (MM), Conference Proceedings*, (Barcelona).

Conflict of Interest Statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

The reviewer, JC and handling Editor declared their shared affiliation.

Copyright © 2018 Damian, Dietz and André. This is an open-access article distributed under the terms of the Creative Commons Attribution License (CC BY). The use, distribution or reproduction in other forums is permitted, provided the original author(s) and the copyright owner are credited and that the original publication in this journal is cited, in accordance with accepted academic practice. No use, distribution or reproduction is permitted which does not comply with these terms.