OpenSense: A Platform for Multimodal Data Acquisition and Behavior Perception

Kalin Stefanov
kstefanov@ict.usc.edu
University of Southern California
Los Angeles, California

Baiyu Huang
baiyu@ict.usc.edu
University of Southern California
Los Angeles, California

Zongjian Li
zongjian@ict.usc.edu
University of Southern California
Los Angeles, California

Mohammad Soleymani
soleymani@ict.usc.edu
University of Southern California
Los Angeles, California

ABSTRACT

Automatic multimodal acquisition and understanding of social signals is an essential building block for natural and effective human-machine collaboration and communication. This paper introduces OpenSense, a platform for real-time multimodal acquisition and recognition of social signals. OpenSense enables precisely synchronized and coordinated acquisition and processing of human behavioral signals. Powered by the Microsoft's Platform for Situated Intelligence, OpenSense supports a range of sensor devices and machine learning tools and encourages developers to add new components to the system through straightforward mechanisms for component integration. This platform also offers an intuitive graphical user interface to build application pipelines from existing components. OpenSense is freely available for academic research.

CCS CONCEPTS

• **Human-centered computing** → Open source software; • **Software and its engineering** → Real-time systems software; • **Computing methodologies** → Machine learning.

KEYWORDS

open source; platform; multimodal; behavior; perception

ACM Reference Format:
Kalin Stefanov, Baiyu Huang, Zongjian Li, and Mohammad Soleymani. 2020. OpenSense: A Platform for Multimodal Data Acquisition and Behavior Perception. In *2020 International Conference on Multimodal Interaction (ICMI '20), October 25–29, 2020, Virtual Event, Netherlands.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3382507.3418832

Figure 1: OpenSense is an open source platform for real-time multimodal data acquisition and behavior perception.

1 INTRODUCTION

Natural and effective human-machine interaction requires machines to understand and produce human-like communicative social signals. Communicative social signals employed in human interaction are multimodal in nature and there has been a substantial interest in the research community for developing platforms for multimodal sensing. OpenSense succeeds MultiSense [21] which was a real-time computational framework, offering flexible and efficient synchronization approaches for context-based nonverbal behavior analysis. MultiSense was developed using the Social Signal Interpretation (SSI) framework [22] which is based on a pipeline architecture. SSI supports real-time recognition of social signals, a large range of sensor devices, filter and feature algorithms, as well as, machine learning and pattern recognition tools. Another similar system is HCI2 [17, 18] which is based on a publisher/subscriber model on top of a message system like Apache ActiveMQ [11]. Similar to the platform described in this paper, Sensei [14] is built on the Microsoft's Platform for Situated Intelligence and is developed primarily to recognize emotions.

In recent years, we have witnessed a significant progress with the development of machine learning techniques targeting many perceptual and control problems. However, building end-to-end

ICMI '20, October 25–29, 2020, Virtual Event, Netherlands

ACM ISBN 978-1-4503-7581-8/20/10...$15.00.
https://doi.org/10.1145/3382507.3418832

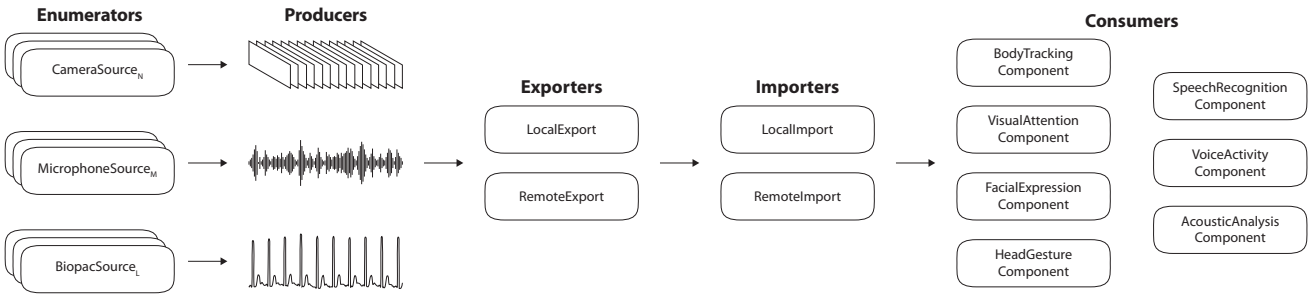

Figure 2: OpenSense consists of mechanisms for device enumeration, producers for acquisition of raw data streams, user interfaces for local and remote export/import of raw data streams, and a diverse set of consumer components for data analysis.

multimodal systems remains a challenging, error-prone and time-consuming engineering task. Such end-to-end multimodal systems should leverage multiple technologies, act autonomously and interact with people in the open world domain. There are numerous challenges to address when developing complex end-to-end multimodal systems including, real-time processing, synchronization and coherent treatment of data streams, and multimodal data fusion.

OpenSense[1] is a platform for real-time multimodal acquisition and recognition of social signals carefully designed to address these challenges. OpenSense supports a range of sensors and machine learning tools and provides an environment that encourages developers to add new sensors and machine learning components through straightforward integration mechanisms. Furthermore, OpenSense offers front-end users an intuitive graphical user interface to build application pipelines from existing components. Unlike the existing multimodal systems, OpenSense is open source, developer- and user-friendly, and provides coherent mechanisms for data synchronization and fusion. Additionally, OpenSense offers mechanisms for distributed computations that are essential for multiagent (multiparty) scenarios or in situations where complex machine learning components compete for computational resources.

The main contribution of this paper is the introduction of an open source platform for multimodal data acquisition and behavior perception called OpenSense. The platform aims to reduce the time-consuming and error-prone engineering task of building end-to-end multimodal systems by offering support for,

- user defined number of input streams;
- user defined number of processing streams;
- distributed computational graphs;
- synchronization and fusion of data streams;
- application design and execution without programming.

2 OPENSENSE

Figure 1 provides a screenshot of the system in action (currently only available on Windows). OpenSense is built on the Microsoft's Platform for Situated Intelligence[2] (\psi) [5], an open source and extensible framework that enables the development of situated integrative-AI systems. Consequently, OpenSense inherits all computational tools provided by the \psi runtime and core libraries, including parallel computing over streams of data, reasoning about time, data stream synchronization, and multimodal data fusion.

2.1 Core

\psi consists of three layers - Runtime, Tools and Components. OpenSense extends the Components layer of \psi by introducing new producers and consumers. An OpenSense producer is a source component that communicates directly with a sensor (*e.g.*, camera, microphone) and streams the captured raw data down the application pipeline. An OpenSense consumer is a consumer component that in the most general case, takes a number of streams as input, *e.g.*, video, and produces a number of streams as output, *e.g.*, facial expression recognition result. An illustration of the current state of OpenSense is given in Figure 2.

2.1.1 Enumerators. Enumerators are OpenSense routines for detection and enumeration of all sensors recognized by the platform. OpenSense provides three enumerators at the moment - camera source, microphone source and biopac source.

The **camera source**, **microphone source**, and **biopac source** enumerators consist of routines that enumerate all camera, microphone, and BIOPAC sensors connected to the machine. The camera and microphone source enumerators are built on top of routines provided by \psi to detect the hardware specifications of the recognized cameras and microphones. The biopac source enumerator is a custom C# implementation and uses the network data transfer (NDT) functionality of BIOPAC to detect the hardware specifications of the recognized BIOPAC devices.

2.1.2 Producers. Producers are OpenSense source components that acquire raw data from the detected sensors and stream it down the application pipeline. OpenSense provides three producers at the moment, one for each enumerator - video stream producer, audio stream producer and physiological stream producer.

The **video stream producer**, **audio stream producer**, and **physiological stream producer** acquire video, audio and physiological data from the camera, microphone and BIOPAC sensors detected by the camera, microphone and biopac source enumerators. These producers are initialized by the user by choosing the desired

[1] https://github.com/intelligent-human-perception-laboratory/OpenSense
[2] https://github.com/microsoft/psi

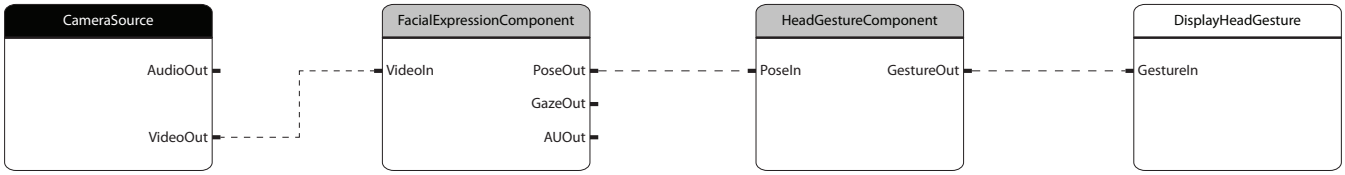

Figure 3: OpenSense Designer enables the definition of custom computational graphs.

sensors and hardware specifications (*e.g.*, framerate, resolution, sampling rate, analog, digital, calculation channels).

2.1.3 Exporters. Exporters are user interfaces that give control over the enumerators and producers. OpenSense users can use the exporters to switch and choose the sensors and set hardware settings that will generate raw data streams in the application pipeline.

2.1.4 Importers. Importers are user interfaces that import raw data streams from the exporters (*i.e.*, enumerators and producers). Importers give control over the consumers that will use the imported raw data streams. OpenSense users can use the importers to switch and choose the consumers that will generate new processed data streams in the application pipeline.

2.1.5 Consumers. Consumers are OpenSense consumer components that take a number of data streams as input and produce a number of data streams as output. OpenSense provides several consumer components that implement access to state-of-the-art libraries for real-time visual and acoustic analysis. Next, we give an overview of the consumer components that are currently implemented in the system.

The **body tracking component** is a custom C# wrapper around an open source C++ implementation of OpenPose[3] [6, 7, 19, 23]. This component can efficiently perform 2D multi-person keypoint detection and 3D single-person keypoint detection based solely on RGB color images. This component consumes video streams from video stream producers, and produces detection results for each frame which can be consumed by latter components in the pipeline. The implementation of this component is based on the OpenPose Unity plugin - an official derivative version of OpenPose for the Unity [13] game engine. Using the default configuration, the body tracking component can reach a near to real-time output frame rate with the help of CUDA [16] capable graphic cards. Better computational performance can also be achieved by reducing the maximum number of tracked people or the scale of the underlying network, without loss in precision.

The **visual attention component** is a custom C# implementation. This component can perform user gaze estimation on a computer screen through personal calibration. This component consumes gaze data streams from facial expression components and combines it with the personal calibration data to produce coordinates on the computer screen. The estimated results for each frame can be consumed by latter components in the pipeline. The functionality is implemented by following the work described in [20] that also allows for head movements.

[3]https://github.com/CMU-Perceptual-Computing-Lab/openpose

The **facial expression component** is a custom C# wrapper around an open source C++ implementation of OpenFace[4] [2–4, 24, 25]. This component can perform facial landmark detection, facial landmark and head pose tracking, facial action unit recognition, gaze tracking and facial feature extraction. This component consumes video streams from video stream producers, and produces detection results for each frame which can be consumed by latter components in the pipeline. Additionally, the base functionality of OpenFace is extended with a facial expression recognition model[5]. Integration of this functionality into OpenSense involves the use of ML.NET[6] with Open Neural Network Exchange[7] to import the pre-trained model into the system.

The **head gesture component** is a custom C# implementation. This component can detect user head nods, head shakes and head tilts. This component consumes head translation and head rotation data streams from facial expression components. The recognition results for each frame can be consumed by latter components in the pipeline. The recognition is achieved through three Recurrent Neural Network (RNN) models, for nods, shakes and tilts where each RNN outputs the probability of each head gesture. The component supports several strategies for fusion of the results and choosing the most probable head gesture. The RNN models are implemented in Keras [8] with TensorFlow [1] backend. OpenSense uses ML.NET with Open Neural Network Exchange to import the pre-trained models.

The **speech recognition component** is a custom C# implementation of three alternative components. Two versions of the component use cloud-based platforms to transcribe speech, Google Cloud Platform [12] and Microsoft Azure [15]. The last version of the component also implements on device speech recognition through the Microsoft Windows 10 native automatic speech recognition functionality. This component consumes audio streams from audio stream producers and the recognition results can be consumed by latter components in the pipeline.

The **voice activity component** is a custom C# implementation. This component detects the presence of speech in the audio stream through the Microsoft Windows 10 native automatic speech recognition functionality. This component consumes audio streams from audio stream producers and the detection results can be consumed by latter components in the pipeline.

The **acoustic analysis component** is a custom C# wrapper around an open source C++ implementation of OpenSMILE[8] [9, 10].

[4]https://github.com/TadasBaltrusaitis/OpenFace
[5]https://github.com/justinshenk/fer
[6]https://github.com/dotnet/machinelearning
[7]https://github.com/onnx/onnx
[8]https://github.com/naxingyu/opensmile

As the rest of the consumer components, the acoustic analysis component can simultaneously consume multiple streams produced by stream producers and generate multiple output data streams. This component consumes audio streams from audio stream producers and the processed data streams can be consumed by latter components in the pipeline. The component can utilize the existing pre-possessing, lower-level, and higher-level feature extraction modules defined through OpenSMILE configuration files.

2.2 Designer

An OpenSense application generally consists of a computational graph that contains a set of components (nodes in the graph), connected via time-aware streams of data (edges in the graph). Most applications use various sensor components (*e.g.*, cameras, microphones) to generate source streams, which are then further processed and transformed by other consumer components in the pipeline. In order to minimize the effort for application development, OpenSense also offers an intuitive graphical user interface, OpenSense Designer, to build application pipelines from existing components. An illustration of the design and definition of a computational graph for a sample application using OpenSense Designer is given in Figure 3.

OpenSense Designer is implemented using React.js[9]. OpenSense Designer is a user interface that consists of two parts - control panel and design panel. From the control panel, users can choose a computational component or a delegate component. A computational component is an OpenSense sensor or consumer component as described previously. A delegate component does not do any computations, this type of component is used for data visualization. Once the desired computational graph components are placed in the design panel, the user can connect the emitters of one component with the receivers of another and export the computational graph to OpenSense. Upon import of the computational graph, OpenSense will verify that the data type of the emitters and the receivers connected in the graph match; then the user can run a custom application based on the defined computational graph.

3 APPLICATIONS

We envision that the platform will be useful in a diverse set of scenarios, here we mention several base use cases. OpenSense can be utilized for **synchronized data acquisition**. OpenSense users can store (synchronized) raw and processed data streams in files. The platform supports both binary data storage (directly importable into \psi Studio and OpenSense) and tools for exporting streams to standard data formats (*e.g.*, .mp4, .wav, .csv). The platform can be used for **synchronized real-time multimodal behavior recognition** and consequently serve as behavior perception component of a **socially interactive system** (*e.g.*, a conversational agent). Additionally, \psi offers a ROS Bridge for integration of ROS components into \psi applications (and OpenSense applications), enabling easy integration with **robotic applications**.

4 ACCESS

OpenSense is freely available for academic research. The code and documentation can be found at https://github.com/intelligent-human-perception-laboratory/OpenSense. Users should respect the licenses of the deployed components.

[9]https://github.com/facebook/react

5 CONCLUSION

We present OpenSense, an open source multimodal behavior sensing platform. We envision that the platform will extend beyond its current state to serve as a component for low-, middle- and high-level sensing capabilities of systems engaged in face-to-face (multiparty) multimodal interactions. The platform will aid researchers during the time-consuming and error-prone engineering task of building end-to-end multimodal systems and enable researchers to carry out smooth and reproducible experiments in multimodal and multiparty interaction. The OpenSense codebase is actively optimized and extended with new components and functionalities and contributions to the code repository are welcome; in the near future we envision that OpenSense will be a community-driven platform. We plan four major directions for future work, including support for more sensor components, support for more consumer components, extended support for distributed computational graphs and implementation of a cross-platform system.

Connecting many consumer components into a computational graph on a single-node server may not meet the fast-growing demand of computational power. One solution is adding the ability to optionally distribute consumer components to nodes in a cluster. This can be achieved by wrapping consumer components using container technology, so that they can be easily deployed and make the system able to scale to the actual demand. In other scenarios, the source components might be physically distributed on different hardware (*e.g.*, robots). OpenSense addresses such cases by providing mechanisms for raw data stream acquisition from sensors on the network (*i.e.*, remote export/import in Figure 2). Distributed computational graphs bring additional challenges related to data compression and encryption that need to be addressed. Resolving these challenges will ensure that the system can run safely and smoothly on the Internet.

OpenSense is currently only available on Windows. The current major issue related to making OpenSense cross-platform is the native code building tool chains of OpenSense's dependencies. We want to upgrade our building tool chain to make the building process easy to manage and suitable for targeting more than one platform.

ACKNOWLEDGMENTS

Research was sponsored by the Army Research Office and was accomplished under Cooperative Agreement Number W911NF-20-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein. We thank the \psi team at Microsoft Research for their continuous support. This research was also supported by Research on Azure Program of Microsoft. We thank Jon Gratch, Skip Rizzo and Rich DiNinni for their support.

REFERENCES

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. http://tensorflow.org/

[2] T. Baltrušaitis, M. Mahmoud, and P. Robinson. 2015. Cross-Dataset Learning and Person-Specific Normalisation for Automatic Action Unit Detection. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*. 1–6.

[3] T. Baltrušaitis, P. Robinson, and L-P. Morency. 2013. Constrained Local Neural Fields for Robust Facial Landmark Detection in the Wild. In *Proceedings of the IEEE International Conference on Computer Vision*. 354–361.

[4] T. Baltrušaitis, A. Zadeh, Y. C. Lim, and L-P. Morency. 2018. Openface 2.0: Facial Behavior Analysis Toolkit. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*. 59–66.

[5] D. Bohus, S. Andrist, and M. Jalobeanu. 2017. Rapid Development of Multimodal Interactive Systems: A Demonstration of Platform for Situated Intelligence. In *Proceedings of the ACM International Conference on Multimodal Interaction*. 493–494.

[6] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. 2019. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[7] Z. Cao, T. Simon, S-E. Wei, and Y. Sheikh. 2017. Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[8] François Chollet et al. 2015. Keras. https://keras.io.

[9] F. Eyben, F. Weninger, F. Gross, and B. Schuller. 2013. Recent Developments in OpenSmile, the Munich Open-Source Multimedia Feature Extractor. In *Proceedings of the ACM International Conference on Multimedia*. 835–838.

[10] F. Eyben, M. Wöllmer, and B. Schuller. 2010. OpenSmile: The Munich Versatile and Fast Open-Source Audio Feature Extractor. In *Proceedings of the ACM International Conference on Multimedia*. 1459–1462.

[11] The Apache Software Foundation. 2020. *Apache ActiveMQ*. http://activemq.apache.org/

[12] Google. 2020. *Google Cloud Platform*. https://cloud.google.com

[13] J. Haas. 2014. A History of the Unity Game Engine. (2014).

[14] D. J. McDuff, K. Rowan, P. Choudhury, J. Wolk, T. Pham, and M. Czerwinski. 2019. A Multimodal Emotion Sensing Platform for Building Emotion-Aware Applications. *CoRR* abs/1903.12133 (2019).

[15] Microsoft. 2020. *Microsoft Azure*. https://azure.microsoft.com

[16] Nvidia. 2020. *CUDA*. https://developer.nvidia.com/cuda-zone

[17] J. Shen and M. Pantic. 2009. A Software Framework for Multimodal Human-Computer Interaction Systems. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*. 2038–2045.

[18] J. Shen, W. Shi, and M. Pantic. 2011. HCI2 Workbench: A Development Tool for Multimodal Human-Computer Interaction Systems. In *Proceedings of the IEEE International Conference on Automatic Face and Gesture Recognition*. 766–773.

[19] T. Simon, H. Joo, I. Matthews, and Y. Sheikh. 2017. Hand Keypoint Detection in Single Images Using Multiview Bootstrapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[20] K. Stefanov. 2010. *Webcam-based Eye Gaze Tracking Under Natural Head Movement*. Master's thesis. University of Amsterdam.

[21] G. Stratou and L-P. Morency. 2017. MultiSense-—Context-Aware Nonverbal Behavior Analysis Framework: A Psychological Distress Use Case. *IEEE Transactions on Affective Computing* 8, 2 (2017), 190–203.

[22] J. Wagner, F. Lingenfelser, T. Baur, I. Damian, F. Kistler, and E. André. 2013. The Social Signal Interpretation (SSI) Framework: Multimodal Signal Processing and Recognition in Real-Time. In *Proceedings of the ACM International Conference on Multimedia*. 831–834.

[23] S-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. 2016. Convolutional Pose Machines. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[24] E. Wood, T. Baltrušaitis, X. Zhang, Y. Sugano, P. Robinson, and A. Bulling. 2015. Rendering of Eyes for Eye-Shape Registration and Gaze Estimation. In *Proceedings of the IEEE International Conference on Computer Vision*. 3756–3764.

[25] A. Zadeh, Y. C. Lim, T. Baltrušaitis, and L-P. Morency. 2017. Convolutional Experts Constrained Local Model for 3D Facial Landmark Detection. In *Proceedings of the IEEE International Conference on Computer Vision*. 2519–2528.