A non-invasive social monitoring application for danger situations based on a edge-based Machine Learning solution.

Bilal Moussa Fares
University of Antwerp - imec
Antwerp, Belgium
Bilal.MoussaFares@student.uantwerpen.be

Pietro Manzoni
Universitat Politécnica de Valéncia
Valencia, Spain
pmanzoni@disca.upv.es

Johann M. Marquez-Barja
University of Antwerp - imec
Antwerp, Belgium
Johann.Marquez-Barja@uantwerpen.be

Juan-Carlos Cano
Universitat Politécnica de Valéncia
Valencia, Spain
jucano@disca.upv.es

Carlos T. Calafate
Universitat Politécnica de Valéncia
Valencia, Spain
calafate@disca.upv.es

***Abstract*—In this article, we analyze the use of non-intrusive monitoring sensors to detect dangerous situations in social events, a typical example being sexual harassment situations during parties. Non invasive sensors can be defined as devices that help measuring some specific parameter of interest remaining hidden or un-perceived by the user.**

We propose a solution based on a social application that can, in a non intrusive way, detect sexual harassment situations and generate alerts automatically to the close by people so that detection of these situations can become quick and independent from the person being attacked. Our results are still preliminary but show that the project can certainly be developed into a working system that can help prevent control crimes from happening at public gatherings.

I. Introduction

Danger situations, like fights or sexual harassment attacks, can occur in public gatherings. According to a cross-sectional study that surveyed 1,160 female students to study the role of alcohol in sexual harassment, ninety-five percent of these assaults were committed by someone the woman knew and almost half of these assaults involved alcohol consumption by either the man, the woman, or both [1]. Bianca et al. [2] conducted 16 in-depth interviews with sexual harassment victims to analyze their experience. They argued that the often cross-border culture of music festivals, combined with the site-specific policing practices and the physical characteristics of the specific locations, created unique obstacles to reporting and had a special impact on responding to and trying to prevent sexual violence at music festivals. There is a vast amount of research that shows the under reporting of harassment, e.g., [3] [4]

Non invasive sensors can be defined as devices that help measuring some specific parameter of interest remaining hidden or un-perceived by the user and are being used in various scenarios. In [5] the author discusses a mobile sensing method for mapping the mobility of people in large-scale events using participatory Bluetooth sensing methods. This non-intrusive technology for collecting spatio-temporal data about participant mobility and social interaction uses the functions of a Bluetooth-enabled smartphone carried by the participant. In [6] the authors describe the development of a light, comfortable, and non-invasive wearable system for the evaluation of cervical spine mobility and the measurement of cervical range of motion are important methods used in clinical settings for diagnosis and prognosis of patients with cervical spine disorders.

In this work, we propose a solution based on a social application that can in a non intrusive way detect sexual harassment situations and generate alerts automatically to the close by people so that detection of these situation can become quick and independent from the person being attacked. This application is integrated in a edge based computing architecture that simplify its deployment even in areas with non or low connectivity. More specifically, we use Machine Learning techniques to turn data measured by a smartphone embedded physical sensors into meaningful patterns to predict and analyze human behavior, and in doing so we could raise alerts or generate data to be shared. The paper is organized as follows: Section 2 describes the edge-based architecture where our solution was embedded. Section 3 details the methodology we used and Section 4 the results obtained. Finally, Section 5 provides more insight regarding the parameters used in our process and Section 6 offers the final conclusions.

II. The FUDGE architecture

In this paper, we make use of an edge/fog generic architecture described here [7] that allows the adoption of edge solutions in IoT deployments in poorly connected and resource limited scenarios. In this architecture we integrate, using microservices, an MQTT based system that can collect ingress data, handle their persistency, and coordinate data integration

with the cloud using a specific service called aggregator. The edge stations have a dedicated channel with the aggregator based on LoRa or WiFi.

This architecture is flexible and robust enough to become an alternative for the deployment of advanced IoT services in resource-constrained contexts, allowing the development of solutions based on edge based Machine Learning like [8], [9].

Figure 1 shows the overall architecture of the FUDGE system. Its basic structure is based on a central node called "aggregator" and several edge stations. The edge stations have a dedicated channel with the aggregator.

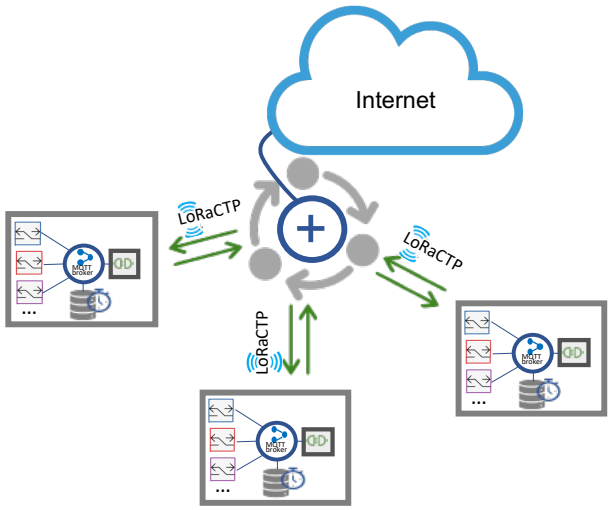

Fig. 1: FUDGE's overall architecture of the proposed system.

The aggregator coordinates the data flow with the edge nodes using a polling approach. The content forwarder is in charge of this task. Basically, at the push stage, the content forwarder sends all the content tagged as Global that are kept stored by the consistency manager. Moreover, in the pull stage, the aggregator sends the content that it kept stored since the previous polling phase. The aggregator returns content whose topics were registered from any service in the edge node. The idea behind this is to give the possibility to local processes to receive both (1) replies to requests sent locally and (2) data from other services in the cloud.

III. Methodology

To reach our goal of non-intrusively detecting dangerous situations for women at parties, multiple steps needed to be taken. We first had to gather sensor data from our data source, which is the smartphone. The data needed to be communicated to our server, to this end, we utilized REST as our software architectural style. Afterwards, the data needs to be stored so we could analyze it later on. Finally, we use the sensor data to train a neural network to detect whenever a person is in danger, this is realized by having the network trained for three states: sitting, moving, and assault.

A. Gathering the data

The deployment we consider is based on a FUDGE node to cover the area we want to control and the smartphones of the users we want to monitor. Including the possibility to use external non-intrusive sensors is also possible. We could for example add an external microphone in rooms, or we could give out accelerometer sensors at the entrance to the party goers. Once a person (or any other party goer) enters the building, their data source would connect to the Internet through the FUDGE node. We designed an app, currently very simple and for Android based devices, to send data to the FUDGE node from the data source. Once the data source was connected and the app launched, sensor data was sent from the client sensors, i.e., the smartphone sensors, to the FUDGE node. This data was communicated using REST. We had one thread in charge on sending the accelerometer, and a second one to send microphone data every 10 seconds, this recording is 5 seconds long.

The data received are stored into a database using the FUDGE architecture. All samples were timestamped and assigned a unique ID so that we could filter the data by user and the timestamp can be used to synchronize the microphone and accelerometer data if necessary. Once this payload is put into a suitable FUDGE format, it is published using MQTT and stored by the FUDGE node in the database.

Fig. 2: Graphical representation of thee accelerometer data.

B. Data preprocessing

After gathering the data, our next step was to analyze it. This step has multiple sub steps. We needed to first assemble the data that we want to analyze, then to split the data into multiple windows before analyzing. Afterwards we could use the Fast Fourier Transform to analyze the frequency content of the data. Once that's done we could extract features from the analysis and finally we could use the features as inputs to the neural network so that it could be trained to detect the danger states.

Using Edge Impulse[1] we gathered and preprocessed data that we collected using our smartphone. We did this in time steps of five minutes for moving around as well as sitting

[1]Edge Impulse is a free development platform for machine learning on edge devices, https://www.edgeimpulse.com

periods. Gathering data for assault was more difficult, to simulate an assault scenario, two people were needed. One would play the role of the assailant, while the other would play the role of the victim. The assailant would get on top of the victim, and the victim would struggle to get out for 30 seconds. During this period, data was gathered from the smartphone. The data was gathered at a frequency of 62.5 Hz.

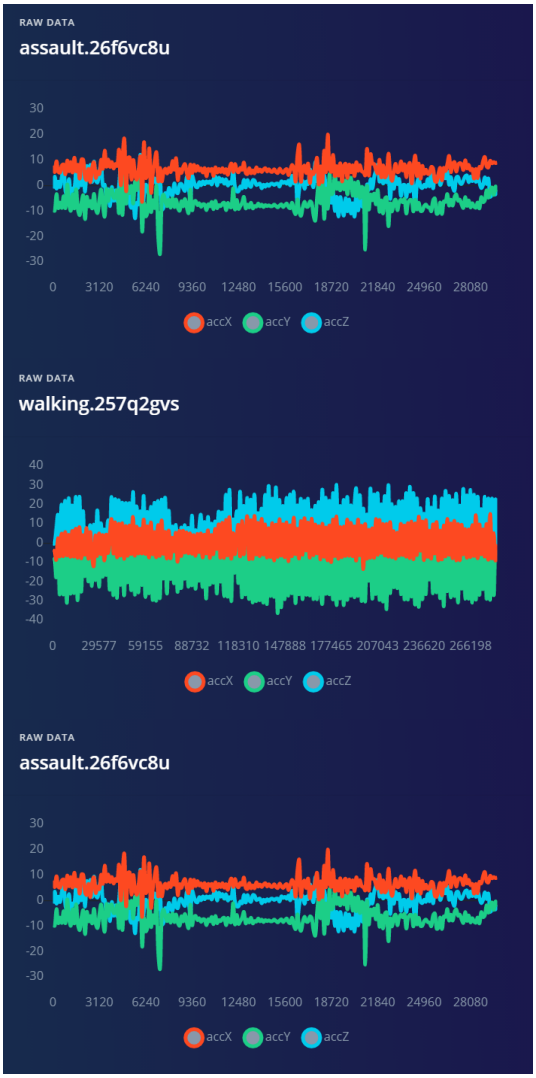

Fig. 3: The gathered accelerometer data, the y-axis contains the amplitude, while the x-axis contains the time in milliseconds

Usually the inputs to a neural network are not simply the raw data. The raw data will serve as a basis to extract features from, these features are the inputs to the neural network. In the case of accelerometer data, this could be, among others, the root mean square of the signal. The features could be extracted using a spectral analysis. The spectral analysis focuses on the frequency content of a signal. To do this, we need to compute the frequency domain from a signal using the Fourier transform, specifically the Fast Fourier Transform. This transformation can be done using multiple parameters. This frequency domain will not be calculated for the entire signal at once, we use windows of a length x milliseconds to analyze the signal using the spectral analysis, this window will be sliding y milliseconds after every time step. To find out which parameters were best, we tested the network out for multiple windows. We used a window size of 1500 milliseconds, 2000 milliseconds and 2500 milliseconds with a sliding window of 100 milliseconds.

The signal underwent multiple transformation. First we filter it to remove unwanted frequencies, the filter that Edge Impulse uses is the Butterworth filter, which has a flat frequency response that would leave our desired frequencies unmodified after applying the filter. Using this filter we can modify the rate at which the frequency is attenuated by modifying the order of the filter (Fig. 4), we opted for a high order so the frequency content that we were interested in would not be attenuated. We could choose to either use a low pass or high pass filter depending on what components we want to remove from the signal. This was not done to erase noise, as the signal itself was not noisy. Rather we did this to analyze whether high or low frequencies are more critical in correctly classifying states. So we chose as a cutoff frequency 3 Hz, which is around half of the maximum frequency that appeared inside our signal. We expected a low pass filter to be less effective, seeing as the movements can change quite fast, as such it would generate a high frequency which should be detected.

The filtered signal underwent a transformation to the frequency domain using the fast Fourier transform. Considering that our sampling rate is 62.5 Hz, every window would have 93 to 156 samples. We usually used 2000 milliseconds for our window, as such we would usually have 125 samples. Our FFT length had around that same amount of points. We chose to use a length of 128 points, seeing as every FFT length demands to be a power of two. We could go higher, however, the resolution would not go higher with the length (Fig. 5). The new points would have their power calculated as an interpolation between the points before and after. This would be useful if we wanted to see the peaks ourselves, however we do not need to do that. Furthermore, a higher length would require more computation time, and seeing as our neural network is accurate enough with just 128 points (see results), it seemed unnecessary and counter productive to further calculate more points.

Out of this transformation we get the spectral power graph as well as the frequency domain graph. The root mean square of all the axis' is calculated as a feature, as well as the peaks in the power graph, the amount of peaks that are calculated is variable and we can choose that ourselves. Once we have the height and frequency of the peaks, we can detect the average power inside power buckets. These buckets are formed by splitting the power graph into multiple parts, the beginning and end frequency for this bucket could be modified. The average power inside these parts was calculated. The end features that we extracted were the root mean square of the signal, the peak frequencies along with their heights and the average power inside the power buckets, this was done for every axis.

All these features could then serve as our input for the neural

network. The network will train itself to link the input features to the given output, which was either sitting, moving, or assault. We do this until there are no more windows available in our signal.

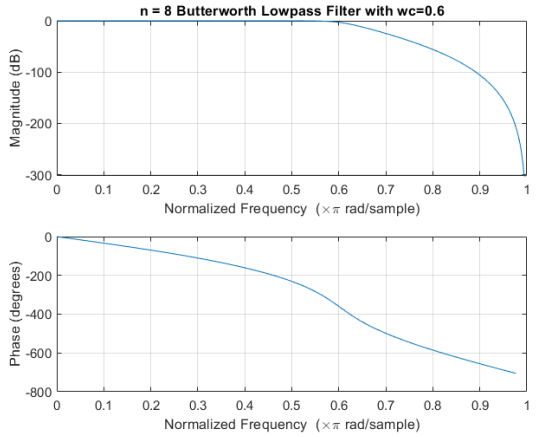

Fig. 4: The magnitude and phase graph of the Butterworth filter, depending on the order of the filter, the frequency that is allowed to pass will be flatter.

Fig. 5: The spectral power graph generated using FFT of variable length, the top figure contains 128 points, the bottom one contains 1024 points. The x-axis contains the frequency in Hz and the y-axis contains the power for that frequency.

IV. Results

After training the neural network using multiple parameters, we have gotten various predictions, some more accurate than others. The only constant seems to be that moving and sitting have not been a problem to predict for the neural network. As expected we had more errors when predicting the assault, the neural network would regularly predict it as simply moving.

prediction results for window of 1500 ms			
	Sitting	Moving	Assault
Low pass	0.955	0.85	0.772
High pass	0.98	0.84	0.82
No filter	0.976	0.857	0.869

prediction results for window of 2000 ms			
	Sitting	Moving	Assault
Low pass	0.961	0.879	0.839
High pass	0.989	0.865	0.805
No filter	0.993	0.929	0.894

prediction results for window of 2500 ms			
	Sitting	Moving	Assault
Low pass	0.971	0.877	0.84
High pass	0.971	0.899	0.836
No filter	0.996	0.913	0.868

We have first computed results for values using a window of 1500 ms. This means that the neural network will expect spectral features as inputs that were calculated using 1.5 seconds of accelerometer data. As expected, the low pass filter was the least successful in classifying data correctly. Sitting and moving have not been a problem in neither option for the filters. The high pass filter did better, with a high general accuracy and 82 percent of assault data correctly classified. The best option here was to not add a filter at all, both high and low frequencies were brought into account and used to calculate the features. This gave the best effect with an accuracy of 86.9 percent for correctly classifying assault moments.

For all the different windows, the only significantly varying result is the sitting result when using a low pass filter. Both the window of size 2500 milliseconds and 2000 milliseconds provided results that were generally a bit better than the window of size 1500 milliseconds.

accuracy for different peaks with window 2000 ms			
	Sitting	Moving	Assault
3 peaks	0.94	0.922	0.894
4 peaks	0.922	0.918	0.897
5 peaks	0.993	0.929	0.894

We can also conclude from the results that having more peaks provides a slightly more accurate prediction level. However this increase seems to be too insignificant to sacrifice processing speed for.

V. The sensing parameters

Our first block was the time series block, here we chose how we would process our data before analyzing it for features: what window we will use and how much we will slide the window every time step. We opted for multiple windows for experimentation purposes. We expected that the accuracy

would correlate with the length of the time window. This was the case going from 1500 milliseconds to 2000 milliseconds. However, above 2000 milliseconds seemed insignificant, this leads me to conclude that around 2000 milliseconds is the ideal window length for our limited data set.

One of the first parameters we had to choose was whether or not we wanted to zero pad our data. This would be especially useful if our window size was larger than our actual data. However this was not the case so we did not need that function. But seeing as it is only used if the window was larger than the data in it, we saw no harm in applying the zero padding function in case the window at edge of our data did not check out anymore even though this would only be one sample out of thousands.

We had around 15 minutes of data, using a sliding window of 1500 milliseconds that moves 100 milliseconds after every time step, this would provide around 9000 samples of data that we can train our network with. Our goal was to have a loss of almost 0.01 or less before we actually validate our network. This was not always the case. Depending on the type of filter we used, we minimized the loss much faster or slower. Using no filter minimized the loss much faster, this is likely due to the features that have much more specific characteristics when no frequencies were left out.

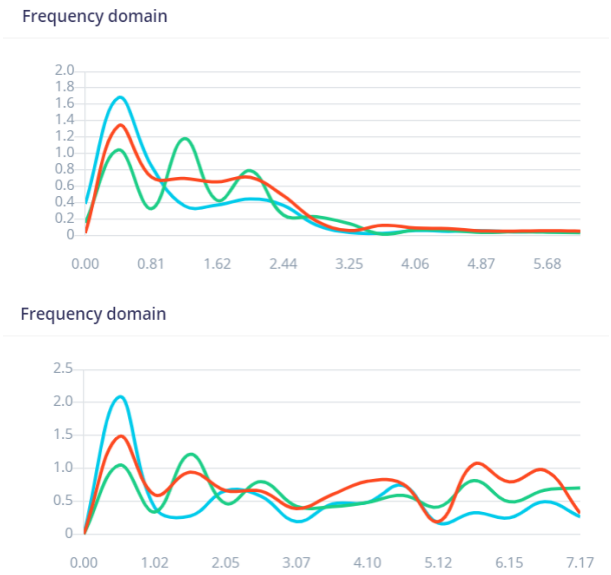

Fig. 6: The frequency domain of a window moving data, x-axis is the frequency in Hz while y-axis is amplitude, top figure shows the filtered signal using Butterworth filter with an order of 6. The figure on the bottom shows that same signal without any filter.

The second block was the spectral analysis block. Here we had to first choose whether or not we wanted to scale our axis'. The important characteristics of our signal was the amplitudes in relation to each other, not the amplitudes by themselves. Changing the scale would not change the proportion of the amplitude of the different frequencies, as such we did not see it necessary to scale the axis, instead we left it as is. We chose to include multiple types of filters and see the difference in results. Filter are usually used to attenuate noise, however, our data was not noisy to begin with. Instead, we were interested in using the filter to see if high frequencies were more present in our data than low frequencies. To that end, we made our cutoff frequency to be three Hz, seeing as our data usually had at most seven Hz as one of the highest frequency. As expected, the high frequencies were more critical in classifying data than the lower frequencies and using all the frequencies available gave an even higher accuracy.

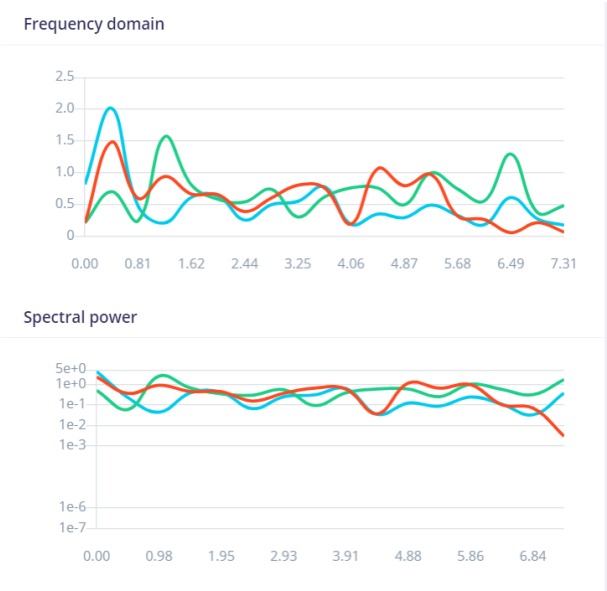

Fig. 7: The frequency domain of a window moving data, x-axis is the frequency in Hz while y-axis is amplitude for frequency domain and power for power spectrum. The FFT size is 128 and no filter was used.

The features that are gathered from the spectrogram were the root mean square of every accelerometer axis as well as the power peaks and the average power in every power bucket. We decided to calculate 5 power peaks, the frequency as well as the height was read out as a feature. Experimentally we decided that 5 peaks' worth of information was enough to have the neural network accurately identify states. However, we didn't want to have the peaks too close to each other so we added a peak threshold of 0.1.

Furthermore, the average power inside of power buckets was also read out as a feature. These power buckets were from 0.1 to 0.5, from 0.5 to one, from one to two and from two to five. We have tried out power buckets that were more uniform, from one to two, two to three, three to four etc... . However they all generally gave the same results, this suggests that the the location of certain quantities of power is less important than the difference between the amount of power in the classified actions in the same power buckets.

The last step was training of the neural network, here we had to decide the amount of training cycles among others, our

goal was to minimize the loss. Experimentally we found that 150 epochs was sufficient to go below a loss of one percent. Seeing as we have a long training cycle, we could afford to have a lower than average learning rate, instead of 0.0005 we went with 0.0003. We designed our neural network with four hidden layers, out of which the first three had 20 nodes, seeing as the input features were 45 nodes, 15 to 20 seemed like a good amount of nodes to put in a hidden layer. In the final layer we reduced it to ten nodes. Furthermore, the features that were extracted were not always clearly distinguishable. The root mean square was the only one where one could see a clear pattern, as for all other features, the pattern was much less clear (Fig. 8). As such we needed more hidden layers to generate more complex polynomials that could fit in the data. This meant more hidden layers, we found that 4 hidden layers were sufficient to correctly fit the data and not be over fitted at the same time.

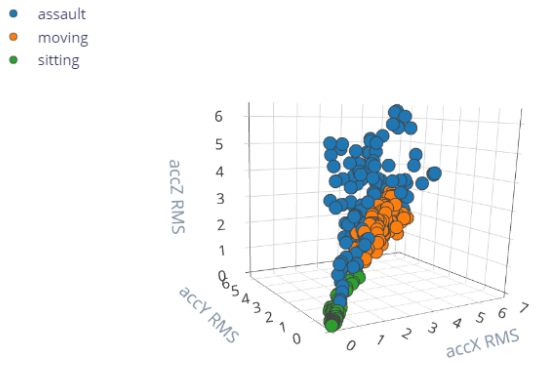

Fig. 8: The root mean square features for the given axis' attributes.

VI. Conclusion

This paper focused on analyzing the use of non-intrusive monitoring sensors to detect dangerous situations in social events, a typical example being sexual harassment situations during parties. Non invasive sensors are defined as devices that help measuring some specific parameter of interest remaining hidden or un-perceived by the user.

We proposed a solution based on a social application that can, in a non intrusive way, detect sexual harassment situations and generate alerts automatically to the close by people so that detection of these situation can become quick and independent from the person being attacked.

We obtained good results in determining the moving and sitting conditions. As expected we had more errors when predicting the assault, the neural network would sometime predict it as simply moving. Our results are still preliminary but show that the project is certainly a good starting point for a useful application. We are positive that, with more effort this first system can be extended into a working system that can help prevent control crimes from happening at public gatherings.

Acknowledgments

This work was partially supported by the "Conselleria de Educación, Investigación, Cultura y Deporte, Direcció General de Ciéncia i Investigació, Proyectos AICO/2020", Spain, under Grant AICO/2020/302 and by the R&D project RTI2018-096384-B-I00, funded by MCIN/AEI/10.13039/501100011033 and "ERDF A way of making Europe".

References

[1] A. Abbey, L. T. Ross, D. McDuffie, and P. McAuslan, "Alcohol and dating risk factors for sexual assault among college women," *Psychology of Women Quarterly*, vol. 20, no. 1, pp. 147–169, 1996. [Online]. Available: https://doi.org/10.1111/j.1471-6402.1996.tb00669.x

[2] B. Fileborn, P. Wadds, and S. Tomsen, "Sexual harassment and violence at australian music festivals: Reporting practices and experiences of festival attendees," *Australian & New Zealand Journal of Criminology*, vol. 53, no. 2, pp. 194–212, 2020. [Online]. Available: https://doi.org/10.1177/0004865820903777

[3] L. Jussen, T. Lagro-Janssen, J. Leenders, C. Logie, and R. Mijdam, "Underreported and unknown student harassment at the faculty of science," *PLOS ONE*, vol. 14, no. 4, pp. 1–10, 04 2019. [Online]. Available: https://doi.org/10.1371/journal.pone.0215067

[4] S. J. Aguilar and C. Baek, "Sexual harassment in academe is underreported, especially by students in the life and physical sciences," *PLOS ONE*, vol. 15, no. 3, pp. 1–18, 03 2020. [Online]. Available: https://doi.org/10.1371/journal.pone.0230312

[5] A. Stopczynski, J. E. Larsen, S. Lehmann, L. Dynowski, and M. Fuentes, "Participatory bluetooth sensing: A method for acquiring spatio-temporal data about participant mobility and interactions at large scale events," in *2013 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops)*, 2013, pp. 242–247.

[6] M. Maselli, E. Mussi, F. Cecchi, M. Manti, P. Tropea, and C. Laschi, "A wearable sensing device for monitoring single planes neck movements: Assessment of its performance," *IEEE Sensors Journal*, vol. 18, no. 15, pp. 6327–6336, 2018.

[7] K. Nakamura, P. Manzoni, M. Zennaro, J.-C. Cano, C. T. Calafate, and J. M. Cecilia, "Fudge: A frugal edge node for advanced iot solutions in contexts with limited resources," in *Proceedings of the 1st Workshop on Experiences with the Design and Implementation of Frugal Smart Objects*. New York, NY, USA: Association for Computing Machinery, 2020, p. 30–35.

[8] Y. Li, Y. Zhuang, X. Hu, Z. Gao, J. Hu, L. Chen, Z. He, L. Pei, K. Chen, M. Wang, X. Niu, R. Chen, J. Thompson, F. M. Ghannouchi, and N. El-Sheimy, "Toward location-enabled iot (le-iot): Iot positioning techniques, error sources, and error mitigation," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4035–4062, 2021.

[9] F. Samie, L. Bauer, and J. Henkel, "From cloud down to things: An overview of machine learning in internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4921–4934, 2019.