Affective Computing & Human-Robot Interaction Group Report

Group "Robocop"

Daniil Gannota
Hussein ElBendak
Connor Daly
Brian Gunawan
Xiaowan Yi
Hanna Ferencz

Contents

1 Aim & Overview

Our goal is to create a discrete classifier for detecting the level of frustration of players in the context of Virtual Reality (VR) shooting games. Given a mixture of modalities consisting of physiological, vocal, and physical signals, we build a multi-channel system that extracts certain features from the collected data and performs classification.

We induce frustration by asking users to play a VR game. The game is special-made, in the sense that it has features designed with the aim of making users frustrated. To collect our data, we use the following sensors:

1. Empatica: To collect physiological signals
2. Notch: To collect data about the movement of the player's moving arm
3. Vive Controller: To allow users to signal when they are frustrated, and to track the rate of trigger presses
4. Microphone: To collect audio signals

There are several use cases for a classifier such as this; frustration in the context of games, if detected correctly, can be a sign of the player being too challenged/not challenged enough, This may give game developers an indication of how to make the game more personalised for every player. Another use case is for the safety of players: game-developers can use this to detect if a player is getting too frustrated at the game, and at a certain point, ask the player if they need a short break.

The steps we have focused on for this project are the following: collecting new data & refining it, fusing the data from multiple modalities, building the recognition model, identifying the discriminative features, labelling the data, and doing a brief evaluation of our final model.

2 Data Collection

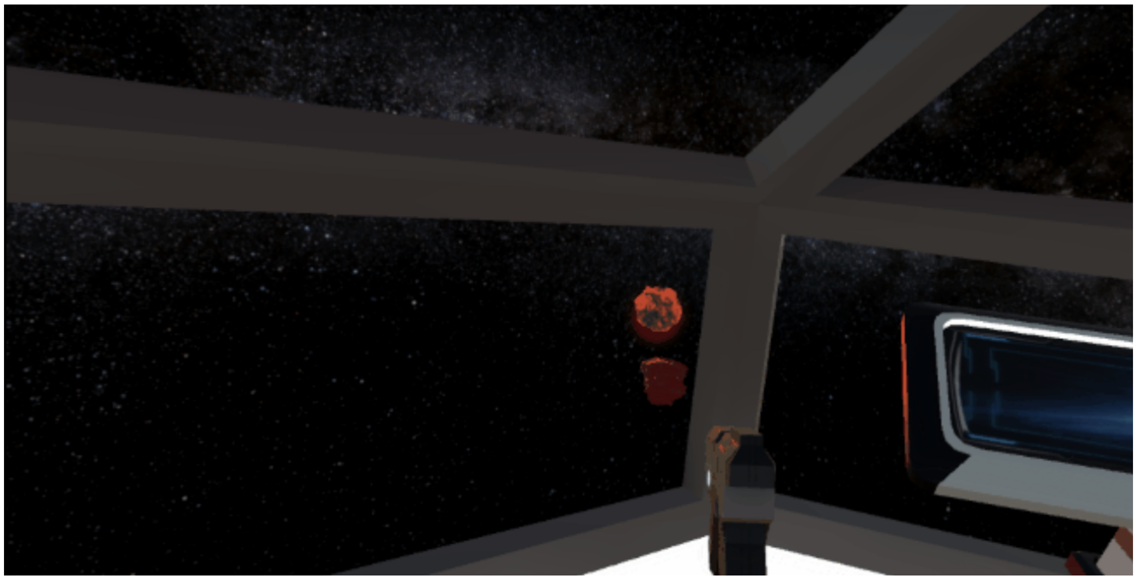

Figure 1: An image of a user playing the VR shooting game we designed. The player can be seen to be holding a gun and aiming at an asteroid out of a spaceship.

The main challenge in our project was designing a virtual-reality game with which we could easily increase the participant's exposure to frustrating stimulus in a controlled, linear manner. We initially trialled a number of different genres of games before coming to the conclusion that it would be easiest to modify the mechanics of a first-person shooter game to increase player frustration.

We developed our own simple game within the game engine 'Unity'. The game was set on a space station, upon which the player must shoot incoming asteroids. The asteroids appeared in waves. After each wave, the number of asteroids and their incoming speed increased exponentially, whilst we also incremented the frustration mechanics. Our frustration mechanics included increasing the trigger delay for the shots fired by the player, as well increasing an angular drift between the direction the player shoots and the actual direction of the shot.

The game was played on the HTC Vive Pro. We made use of Notch motion sensors, an Empatica, the controller the user was playing, and a microphone. Three notch sensors were placed on the participant's dominant arm with which they held the Vive controller, capturing the rotation angle of the arm, as well as the x, y, z coordinates of each sensor, in world-space.

The Empatica was placed on the participant's non-dominant arm, as we found that rapid movement of the sensor often led to noisy results. The sensor recorded heart-rate (HR), galvanic skin response (EDA), blood volume pulse (BVP) and body temperature. The microphone captured the player's vocal response to the game, from which we later extract and correlate player sentiment during game-play. In addition, the Vive Controller was used to log button presses.

The first few minutes of the game had no incoming asteroids and was used to ensure that the participant was in a relaxed state. During this time, we noted a baseline for the individual with regard to physiological modalities. When the game commenced, the participant voiced the word ‘start’, an event was recorded on the Empatica, and the Notch recording was initiated. Several team members were needed to ensure all of these steps were done at the same time, to aid with data fusion later on. To mark the end of an experiment, the Empatica button was pressed as soon as the game ended, the Notch recording was stopped and the player said the word ‘stop’ for the microphone.

We adopted a semi-supervised approach to labelling the data. During game-play, the participant was instructed to press a button on the controller, registering the time at which they felt frustrated. The time recorded was used to later isolate a time window within the trial data. Trigger press frequency throughout the game was also logged.

3 Data Fusion & Machine Learning Techniques

We attempt two different fusion models, and compare their performance. Before describing the different fusion techniques, we give an overview of the pre-processing we perform on our data.

3.1 Data synchronisation

To begin, we had to synchronise the data acquired from our different sensors. The raw data we collected had different formats, frequencies, and time measurements, and thus, we had to begin by aligning and standardising our data.

The raw data collected from the Empatica was presented in the form of absolute time since the sensor started recording. Moreover, the Empatica provided a file which included information about the marked events, using which, we could determine when the game started and ended.

The raw data collected form the Notch was presented in the form of absolute time. Since the Notch sensor was started and stopped manually with the beginning and end of the game, these were matched to the events from the Empatica.

The raw data collected from the microphone was sliced off manually by listening to them and cutting all the voice data before and after the 'start' and 'stop' signals.

Lastly, the data collected from the Vive controller was presented in POSIX time, and so had to be converted to absolute time since the experiment started, to match the data from the rest of our sensors.

3.2 Feature Extraction

3.2.1 Window Size

To extract particular data points from our masses of data, we used the 'Frustrated Button' presses as our guide.

For each 'Frustrated Button' press in our data, we take a window of x seconds around the press ($\frac{x}{2}$ seconds before and $\frac{x}{2}$ seconds after), and extract data from all our different modalities according to that window.

3.2.2 Empatica

For the HR, BVP, EDA and temperature, we extracted the data measured over the window size. The amount of data points this equalled to differed between each one, since Empatica uses a different frequency to measure each of the HR, BVP, EDA and temperature.

For each of the HR, BVP, EDA, and temperature and an appropriate window of data, we then normalise (explained more in 3.3) our list of data points, and proceed by extracting the following and using as features:

1. The mean of each over the window
2. The variance of each over the window
3. The maximum of each over the window
4. The minimum of each over the window

3.2.3 Notch

For Notch data, we extract only the trajectories of each marker. The sensors record at a rate of 64 *Hz*, for each window of 10 seconds we therefore have approximately 640 frames of positional data for each marker. Following the example of Glowinski et al.[1] we use *'jerk'* as a measure of participant motion smoothness. Movement jerk is defined as the 3^{rd} derivative of motion We first compute the speed of the notch between each frame and then compute the acceleration between each velocity in the window. From the computed set of accelerations, we are able to compute a cumulative jerk score across the entire window by summing the jerk between each acceleration.

3.2.4 Voice

The aim of this modality was to pick on players' "grunts" and angry comments, to reflect on their frustration levels.

To identify human speech, Mel Frequency Cepstral Coefficients (MFCCs) were used for pre-processing of the audio.

Mostly, MFCC's main focus is on speech detection and recognition. However, we were researching the changes in voice and the intensity of comments/"grunts". Therefore, to measure the changes in players' voice, we linked the audio to the button presses. The audio corresponding to each button press' window is used for the analysis. We extracted 5 MFCC features from the window of audio.

Additionally, to improve the accuracy of the results, these features are differentiated twice. Lastly, we sum all of the values within each window and output this as a single number representing the window. Hence, the number represents the intensity of the audio and the loudness, both of which may indicate the levels of frustration.

Overall, more intense comments/"grunts" were producing a bigger absolute number, and if a window did not have any speech detected from the player, then the absolute value of the window was closer to 0.

3.2.5 Trigger frequency

Over each window, we also extracted the number of times the trigger button was pressed. The intuition behind this is that as the user gets more frustrated, they may start spamming the trigger button.

3.3 Normalisation

To be able to compare the physiological modalities such as heart rate and blood volume pulse between different people, we normalise our data points on a person-to-person basis.

For each person and measurement, we extracted the baseline by obtaining a window of data for 5 seconds at the beginning of the experiment, we then averaged the measurements over these 5 seconds, and used these as our baseline.

After obtaining our baselines, we use this to normalise the data points we extracted before they're passed on to the machine learning model. For example, after extracting a particular data point for heart rate $x_{i,j}$ for user i and button press j, and calculating the heart rate baseline b_i for user i, we calculate our normalised data point as $\bar{x}_{i,j} = x_{i,j} - b_i$.

3.4 Learning then Fusing

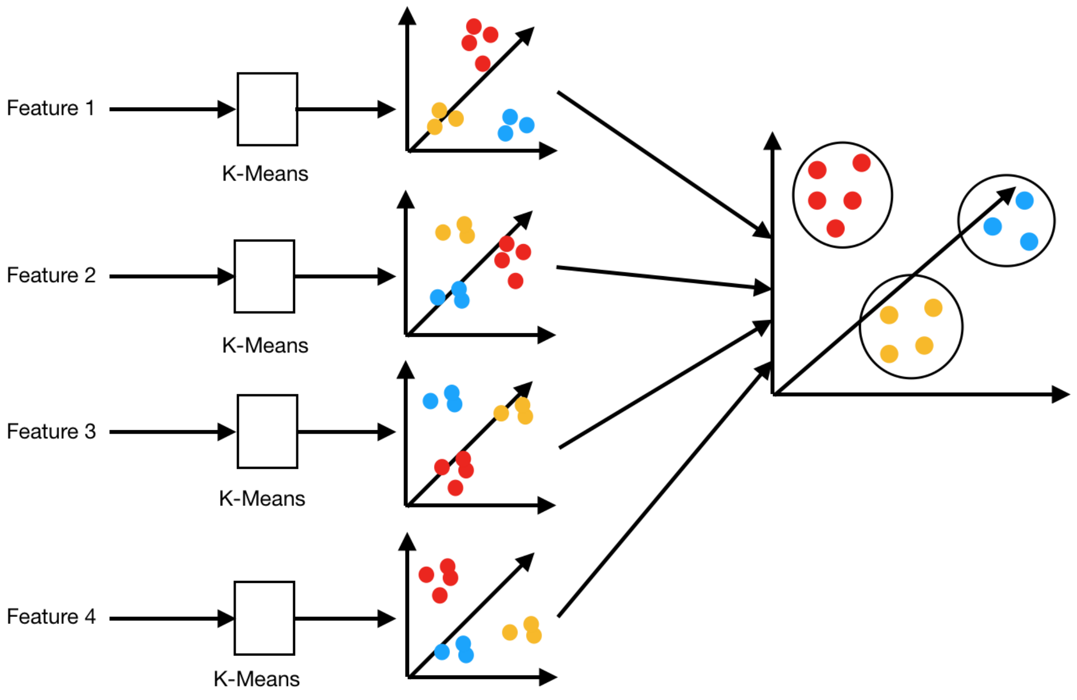

Figure 2: Performing K-Means Clustering on our different features, then fusing the results.

In this approach, we attempt to cluster each feature separately, and then make use of an ensemble to combine the final data together.

Each individual model is a K-Means algorithm dependent only on the measurements from one feature (using Euclidean distance). Each feature will, in its respective K-Means algorithm, assign a data-point to 1 of 3 clusters. Obviously, ensembling is not straightforward as there are no ground truths, therefore to map each of the 3 clusters in each model to a common identity across all models, we examine which data-points are most frequently clustered with other data-points.

We can then generate 3 ensembled clusters, based on which data-points most frequently co-occur.

3.5 Fusing then Learning

In this fusion model, we start by extracting all the features for our different data points per every participant. After performing all the necessary pre-processing, we create a feature vector $w \in \mathbb{R}^d$, where d is our number of features.

At this stage, we experimented with different models and settings using all of our features. As an overview, we attempt the following:

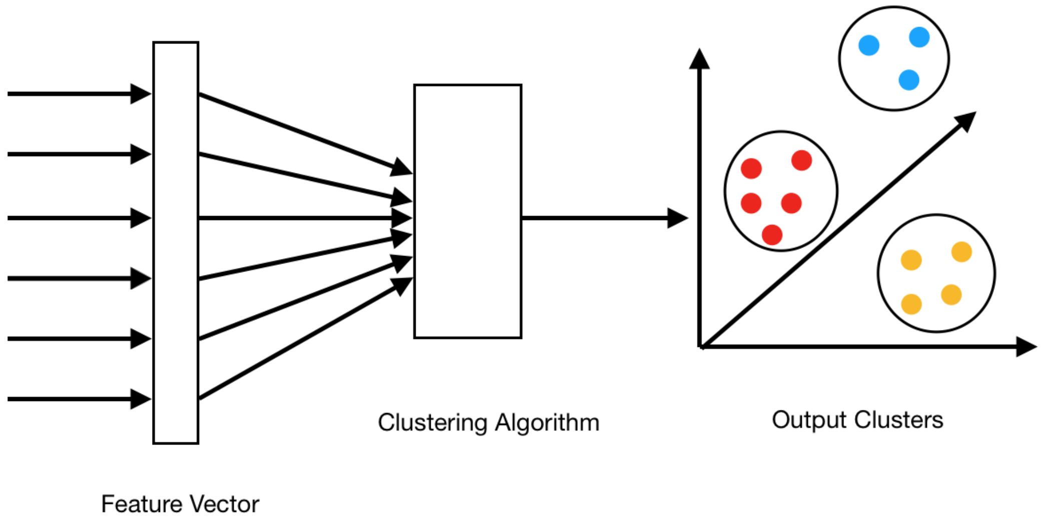

Figure 3: Fusing our features, and then performing K-Means Clustering.

1. We experiment with training the model using different Unsupervised Learning algorithms, such as K-Means and Gaussian Mixture.

2. We experiment with training the model using a different number of clusters.

3. We attempt to use different window sizes to extract our features.

The metric used for clustering for all of these models was the Calinski Harabez Score (defined as the ratio between within-cluster dispersion and between-cluster dispersion).

4 Evaluation of Models & Features

4.1 Learning then Fusing Model

Initially we trialled an ensemble methodology using 4 clusters, passing in non-frustrated data in addition to data-points corresponding with times when players pressed the frustration button. However, during the ensemble voting, this resulted in highly uneven clusterings, with an empty cluster, and a majority cluster containing 36 of the datapoints. Reducing the number of clusters to 3, this results in a more evenly distributed clustering from the ensemble.

To explore the relative role each feature plays in the ensemble with 3 clusters, we first compute the simple euclidean distance between each of the centre points from the clusters formed in the single feature models. The values for each feature are normalised so that the resulting distances are comparable.

The 5 models with the greatest inter-cluster distance were: EDA variance, Heart-Rate variance, BVP minimum, BVP average, and BVP minimum. The 3 models with the smallest inter-cluster distance were: Heart-Rate average, Heart-Rate maximum and voice data. We can see the modalities which are most distinct in terms of clustering, and those that are the least.

To explore which modalities are most correlated with the final output of the final cluster ensemble, and therefore less prone to individual noise, and more likely to give a reliable indicator of cluster group, we compute a difference score for each modality. Using the clusters predicted by the model that only uses an individual modality, and the clusters predicted by the ensemble, for each datapoint, we compare the clusterings. If the clustering is different we increase the difference score for that modality. The 7 modalities with the lowest difference score are EDA minimum, Heart-Rate average, Heart-Rate maximum, Heart-Rate minimum, BVP average, EDA average, and EDA maximum. The 3 modalities with the greatest difference score are Heart-Rate variance, jerkiness of upper arm, and BVP minimum.

The average normalised feature values (for top 7 most correlated features) within the clusters produced by the ensemble are:

	EDA min	**HR avg**	**HR max**	**HR min**	**BVP avg**	**EDA avg**	**EDA max**
Cluster A	*0.12*	10.7	11.5	9.8	*-2.4*	*0.20*	*0.30*
Cluster B	**0.24**	**12.2**	**12.9**	**11.6**	-0.9	0.29	0.35
Cluster C	0.23	*-1.74*	*-0.47*	*-1.7*	**0.83**	**0.33**	**0.50**

Table 1: Average values of top 7 features across resulting clusters. Maximum value of each feature is in **bold**, and minimum value in *italics*

4.2 Fusing then Learning Model

Since we experimented with different Machine Learning models and settings, we begin by identifying the best performance model setting using the Calinski Harabaz score (higher scores can be interpreted as better clustering in terms of how good the separation is). We find that K-Means with a window size of 4 seconds (2 seconds before and 2 seconds after) actually performs best.

Next, we would like to extract the discriminative features of our best model. For each feature in the training data, we temporarily leave that feature out and train the model using the rest of

Model	Calinski Harabaz Score
KM with 3 clusters, window size 10	10.44
KM with 3 clusters, window size 4	16.42
KM with 4 clusters, window size 10	10.81
KM with 4 clusters, window size 4	14.13
GM with 3 clusters, window size 10	9.35
GM with 3 clusters, window size 4	12.04
GM with 4 clusters, window size 10	9.89
GM with 4 clusters, window size 4	13.09

the features. We compute the corresponding Calinski Harabez scores, the smaller the score is, the more positive influence the removed feature has over the clustering performance, meaning that the feature is important. This was done iteratively over all the features, removing the feature with the minimum score after every iteration and recording it as an important feature.

We proceed by analysing the best top N important features to train the model, and find that N=7 gives the best score. This is illustrated in Figure 4.

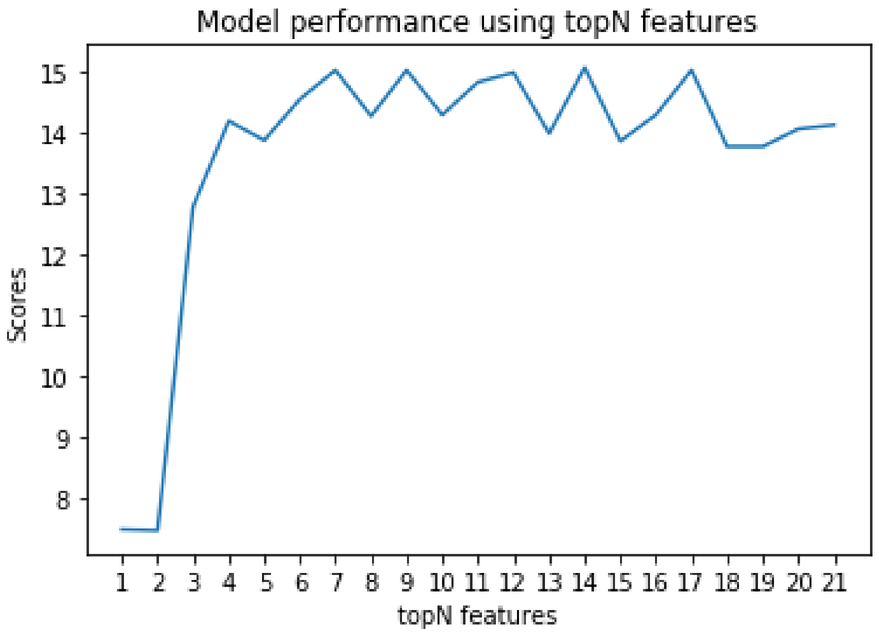

Figure 4: Performance using top N features

We train the final fusion model with the top 7 features, and then look at the average values over each cluster in Table 2.

	EDA Min	**EDA avg**	**FA Jerk**	**Temp Max**	**Audio**	**BVP Min**	**HR avg**
Cluster A	**0.77**	**0.64**	0.26	*0.36*	0.40	**0.92**	0.31
Cluster B	0.14	0.09	*0.064*	0.54	**0.71**	*0.81*	**0.63**
Cluster C	0.16	0.09	0.36	0.39	*0.24*	0.90	0.37
Cluster D	*0.12*	*0.06*	**0.58**	**0.92**	0.35	0.90	*0.27*

Table 2: Average values of top 7 features across resulting clusters. Maximum value of each feature is in **bold**, and minimum value in *italics*

5 Data Labelling & Model Evaluation

5.1 Self-Reporting

Before the experiment started, users were instructed that there is a button on the Vive controller which maps to 'I'm Frustrated'. They were told to simply press the button whenever they feel frustrated at the game.

5.2 Using the Literature

The self-report method only allowed for indications as to when the users were feeling frustrated, but were not indicative as to the user's level of frustration. As a result, we looked at the average of the data points' features in each cluster, and referred back to the literature to label these clusters accordingly. We will focus on the dominant features extracted in our two main models, presented in Tables 1 and 2. Looking at the literature, we can see that:

1. EDA - Higher EDA implies higher level of frustration [2]
2. Heart Rate - Higher HR implies higher level of frustration [2]
3. BVP - Used to calculate HR so assumed to have same analysis (Higher BVP implies higher level of frustration)
4. Audio - Raised voice implies higher level of frustration [3]
5. Temperature - Higher skin temperatures are correlated with more negative emotions [4]
6. Foreram Jerk - Closest thing in the literature is relating fidgeting to frustration [5] (More fidgeting means a higher level of frustration, we will assume this is true for arm jerkiness as well)

Our findings in the literature makes the task of labelling each cluster a fairly systematic one, since we have an agreement that all these features should be maximised at the highest level of frustration, and minimised at the lowest level.

5.3 Labelling our Clusters & Evaluating our Models

First, looking at the results from the model where we fuse then learn (Table 2), we find that the results are mostly inconclusive, where the maximum and minimum values of features co-occur within the same clusters, making it impossible to define a level of frustration to each one.

Next, we look at the results from the model where we learn then fuse (Table 1), if we only look at the top four features, we can very easily, and with high confidence, determine that Cluster B corresponds to the highest level of frustration, Cluster A corresponds to the medium level of frustration, and cluster C corresponds to the lowest level of frustration. However, this pattern does not extend to the rest of the top 7 features (BVP avg, EDA avg, EDA max).

Our labelling rules out the fusing then learning models due to the feature values within each cluster indicating contradicting mappings to frustration, and shows that our learning then fusing model performs fairly well, if we only rely on EDA and HR as a measure of frustration.

References

[1] D. Glowinski, N. Dael, A. Camurri, G. Volpe, M. Mortillaro, and K. Scherer, "Toward a minimal representation of affective gestures," *IEEE Transactions on Affective Computing*, vol. 2, no. 2, pp. 106–118, 2011.

[2] A. Drachen, L. E. Nacke, G. Yannakakis, and A. L. Pedersen, "Correlation between heart rate, electrodermal activity and player experience in first-person shooter games," in *Proceedings of the 5th ACM SIGGRAPH Symposium on Video Games*, Sandbox '10, (New York, NY, USA), pp. 49–54, ACM, 2010.

[3] J. Ang, R. Dhillon, A. Krupski, E. Shriberg, and A. Stolcke, "Prosody-based automatic detection of annoyance and frustration in human-computer dialog," *In ICSLP–2002*, pp. 2037–2040, 2002.

[4] S. C. Müller and T. Fritz, "Stuck and frustrated or in flow and happy: sensing developers' emotions and progress," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1, pp. 688–699, IEEE, 2015.

[5] A. Kapoor, W. Burleson, and R. W. Picard, "Automatic prediction of frustration," *International journal of human-computer studies*, vol. 65, no. 8, pp. 724–736, 2007.