DETECTING USER FRUSTRATION FROM SMARTPHONE SENSORS: A MULTIMODAL CLASSIFICATION APPROACH

by

ESTHER VASIETE

B.S., Telecommunication Engineering

Universitat Autònoma de Barcelona, 2013

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirement for the degree of

Master of Science

Department of Electrical Engineering

2015

Certificate of Completion

This thesis entitled:

Detecting User Frustration from Smartphone Sensors: A Multimodal Classification Approach

written by Esther Vasiete

has been approved for the Department of Electrical Engineering

University of Colorado at Boulder

____

Tom Yeh

____

Shaun Kane

____

Peter Mathys

Date_______________

The final copy of this thesis has been examined by the signatories, and we Find that both the content and the form meet acceptable presentation standards Of scholarly work in the above mentioned discipline.

IRB protocol # 13-0627

ABSTRACT

Vasiete, Esther (M,S., Electrical Engineering [Department of Electrical, Engineering])

Detecting User Frustration From Smartphone Sensors: A Multimodal Classification Approach

Thesis directed by Professor Tom Yeh

Most smartphone applications are unaware that users feel frustrated by a bug, an error, or a usability problem. Could a smartphone be “smart” enough to detect that its user became frustrated by something? A pilot study was conducted as proof of concept, showing that sensor readings collected in the background from a smartphone could be used to detect user frustration after the onset of a frustrating event due to a bug or a usability problem. Twenty-one participants were asked to perform a series of multitasking tasks, during which errors were purposely introduced to frustrate them. Sensor data, including motion, touch, and camera, were collected and used to train a binary classifier that is able to detect frustration with reasonable accuracy when merging data from different modalities (motion sensors and touch gestures).

CONTENTS

TABLES

FIGURES

CHAPTER 1

INTRODUCTION

Motivation

Frustration occurs with the inability to change or achieve something, and the prevention to fulfill one's wishes or expectations. Physiologists and cognitive scientists have been interested in the study of frustration, as it contributes on human behaviors that immediately follow a frustrating event, and have used functional magnetic resonance imaging (fMRI) to identify the areas of the brain that are highly correlated with the presence of frustration [1]. Researchers have also found that humans often express and release frustration towards inanimate objects, e.g., slamming a door or pressing the keys of a computer keyboard with higher intensity [2, 3].

Novel user interfaces that assist the user when he/she is feeling frustrated are susceptible to emerge with the ability to automatically detect user's frustration when using a smartphone. For instance, user's frustration could expose usability issues in *apps* and enable them to make necessary adaptations. Other applications include intelligent tutoring systems that could react to a student's frustration to provide better learning experiences, as it was found in [4] that frustration is inversely correlated with the student's learning gain.

One challenge in detecting frustration is that it is a dynamic state that evolves over time, making it hard to find a suitable ground-truth criterion and validation system. In the case of frustration, many rely on personal interviews or self-reporting about the subject's feelings, but these reports can be highly misleading for several reasons. First, timing information is lost because frustration is usually reported at the end of the task. Second, the user may forget the moment he/she became frustrated or might not even realize that he/she was frustrated after the "heat of the moment". It is also worth noting that the reliability of the reported status is greatly impacted by the subject's personality and the selection and style of the questions themselves. Lastly, the subject may not be comfortable enough with reporting self-frustration and might disguise the answers.

Previous researchers have focused on facial expressions to detect different affective states, frustration among them [5,6] with rather good accuracy. However, numerous current studies use overexpressed prototypical emotions and have not been tested on an ordinary environment [7, 8]. Moreover, facial expressions are not always related to analogous unequivocal emotions, as facial expressions are usually ambiguous and very dependent on the person's character. For instance, it was shown in [9] that some people smile during frustrating events, making it one of the most difficult states to detect from visual cues [10].

Behavioral modalities have previously been used to detect frustration by extracting different types of sensorial data. This includes physiological signals such as electrodermal response, blood volume pressure [11] and skin conductance [3]. Other sensors include an emotional mouse to measure the heart rate, skin temperature, and finger pressure [12]. The intrusiveness and the extra equipment required make these procedures very inconvenient in most situations.

To the best of my knowledge, there is no previous work on the detection of user frustration on smartphones; thus, this research could have a significant impact on the HCI community with the emergence of applications that assist the user for a better learning experience in intelligent tutoring systems or the instantaneous feedback of a usability bug in *app* testing systems, among other possible applications.

Related work

In the literature on automatic frustration detection, two methodologies are often used to induce frustration. The first method relies on a timeline where a subject is required to perform a simple task with increasing difficulty over time (e.g. harder math problems [12], a Tetris game with more difficult-to-fit bricks [13] or the Towers of Hanoi puzzle, which inherently becomes harder [3]). However, this method has two major drawbacks. A subject may not necessarily feel frustrated when a task gets harder; there might be an expectation that tasks or games are getting harder in order to retain challenge and engagement. The second drawback is timing control, as even if a subject does in fact begin to feel frustrated, it is hard for researchers to identify the precise onset of such feeling for the purpose of training a classifier. Experiments that follow this timeline have relied on subject self-reporting of frustration during the experiment or after its completion, leading to incomplete and biased data.

The second method [11, 14] attempts to overcome these problems by introducing a frustrating event at a known time t_f following the timeline below:

a. Subject completes task *x*.

b. A frustration event occurs at time t_f.

c. Subject performs another instance of task *x*.

If the event is sufficiently frustrating, it could be assumed that the subject became frustrated at time t_f after such event. In this framework, the subject performs the same task before and after t_f, providing a proper setting of the control and frustration conditions.

Although no prior work has been yet conducted on the detection of frustration on smartphones, it has been shown that touch gestures can be used to model behavioral analysis. For instance, it has been demonstrated that a subject's touch retains individual traits usable for authentication purposes [15]. Accelerometer and gyroscope sensors have been used to achieve similar work [16, 17]. Performing nearly real-time facial emotion recognition techniques in smartphones is feasible [18]. However, due to the nature of the front camera recording, some issues with face and emotion recognition have been encountered [19], in addition to the inherent problems of detecting emotions from physical appearances [9, 10].

This thesis presents a novel study on the detection of frustration on a smartphone user through multimodal sensor and touch data recorded in the background while the user is performing any task.

CHAPTER 2

EXPERIMENT DESIGN

An experiment to deliberately frustrate subjects in a pilot study was designed based on the second timeline scheme (i.e. inducing a frustrating event at known time t_f). The experiment consists of a rather simple smartphone game (developed for Android) that contains two programmed frustrating events disguised as system or network errors. The main objective of the game, as told to each subject, was to analyze the subject's multitasking skills for a cognitive science research project.

The game has four stages. In each stage, a two-minute video clip is played displaying various cutscenes extracted from the Hollywood Human Actions Dataset [20]. Each video clip contains around 40 different cutscenes from Hollywood movies, each displaying a particular action. Each video clip is also accompanied with a song. The multitasking game consists of looking out for specific actions and listening for specific words in the background song.

Specifically, the subject's task is to tap on the right side of the screen when they hear the target word in the background song and on the left side of the screen when they see the target action in the video. For instance, the first clip is about "kissing" and is made of 17 short segments of people kissing among 34 other random short cutscenes (which include any other action). While doing so, the song "Kiss Me" by *Sixpence None The Richer* plays in the background, which contains 7 utterances of the word "kiss". The three remaining clips are about holding a phone, handshaking, and getting out of a car. The instructions of the game, as shown to the subjects, can be seen in Fig. 1. A more detailed description of each stage is presented in Table 1.

Subjects are able to learn their score (Fig. 2a) after each stage to provide feedback on their performance and incentivize self-improvement and engagement. A final score (Fig. 2b) is computed at the end of the four stages, and after they have submitted a survey form (Fig. 3). Subjects are asked to provide personal information as well as an impression and feedback about the multitasking game in the survey. All the fields of the form are mandatory and subjects cannot leave the application without entering all the answers.

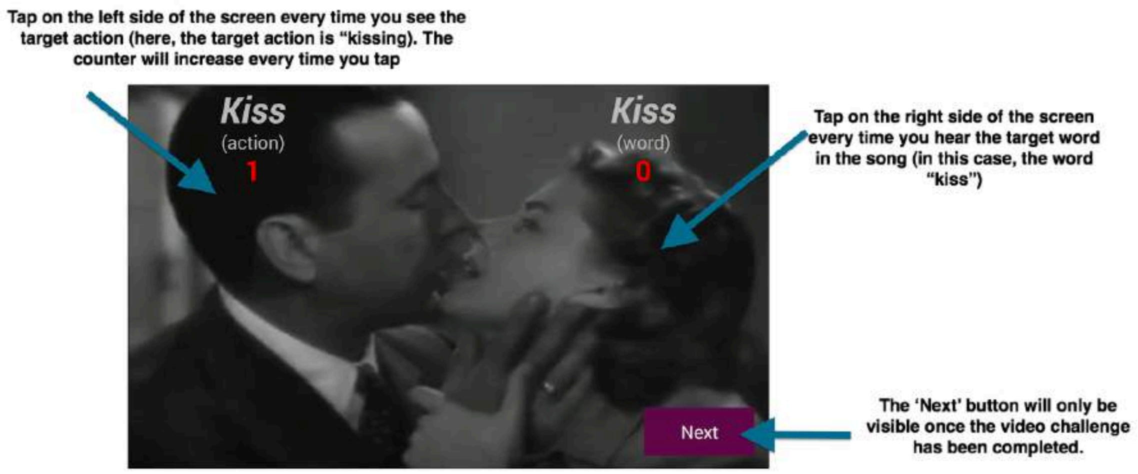
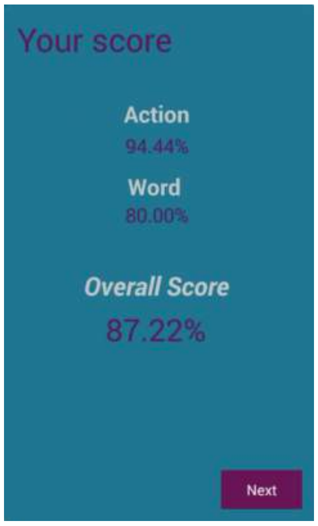

Tap on the left side of the screen every time you see the target action (here, the target action is "kissing). The counter will increase every time you tap
Kiss
(action)
1
Kiss
(word)
0
Tap on the right side of the screen every time you hear the target word in the song (in this case, the word "kiss")
Next
The 'Next' button will only be visible once the video challenge has been completed.

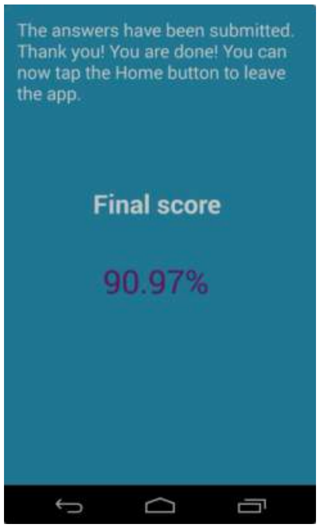
Figure 1. Instructions of the multitasking game, as shown to the subjects.

Stage	Target Action	# target scenes	# random scenes	Target Word	# instances of target word in song	Background song
1	Kiss	17	34	Kiss	7	"Kiss Me" by *Sixpence None The Richer*
2	Hold a phone	18	31	Phone	15	"Banana Phone" by Raffi
3	Handshake	14	25	Shake	43	"Shake your Booty" by *KC & The Sunshine Band*
4	Get out of car	14	32	Everywhere	12	"I've been everywhere" by *Johnny Cash*

Table 1. Details of each stage of the multitasking game.

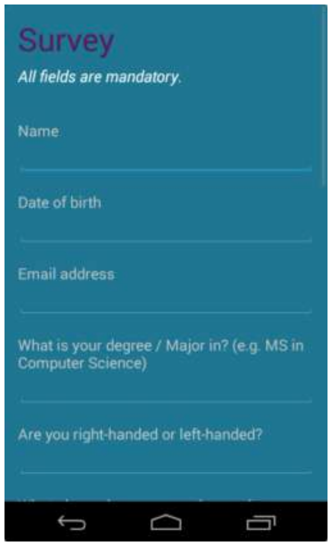


a)


The answers have been submitted. Thank you! You are done! You can now tap the Home button to leave the app.

b)

Figure 2. a) Score after each stage, b) Final score based on all four stages.


Survey

All fields are mandatory.

Name

Date of birth

Email address

What is your degree / Major in? (e.g. MS in Computer Science)

Are you right-handed or left-handed?

Figure 3. Survey form screenshot

Frustrating events (FEs)

During the experiment, two frustrating events (FE1, FE2) were programmed. The first event (FE1) occurred in stage 3, which is about "handshaking". The clip includes 14 short segments of people handshaking and 43 instances of the word "shake" in the background song "Shake your Booty". A screenshot of stage 3 is shown in Fig. 4.

After completing the first two stages subjects would be familiarized and engaged with the game. About 50 seconds into stage 3, a fake network error occurred (Fig. 5a), interrupting the subject's concentration on the task, and followed by a fake loading dialog (Fig. 5b) for 10 seconds. Right after that, the video froze, and the sound was muted for a short period before the game appeared to normally function.

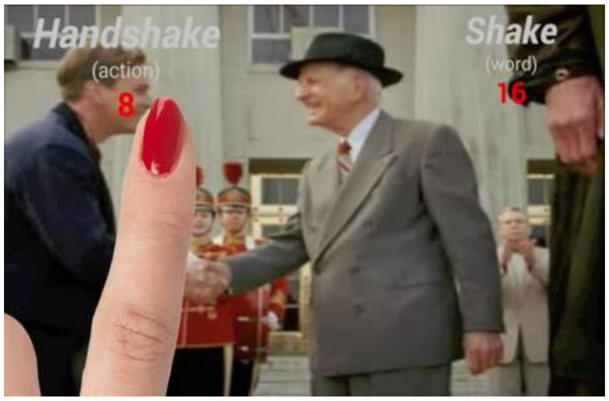
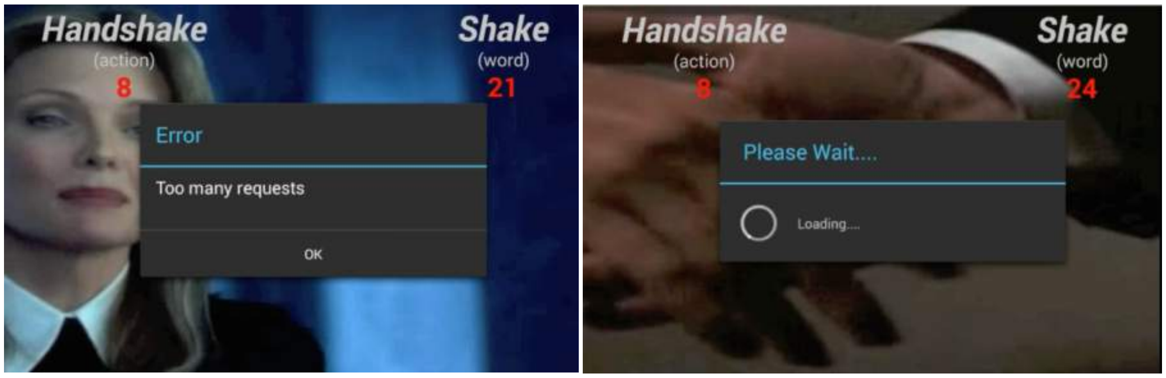


Figure 4. Screenshot of multitask game at stage 3. Subjects need to tap on the left side of the screen when they see a handshake action and tap on the right side of the screen when they hear the word "shake".



Error

Too many requests

OK

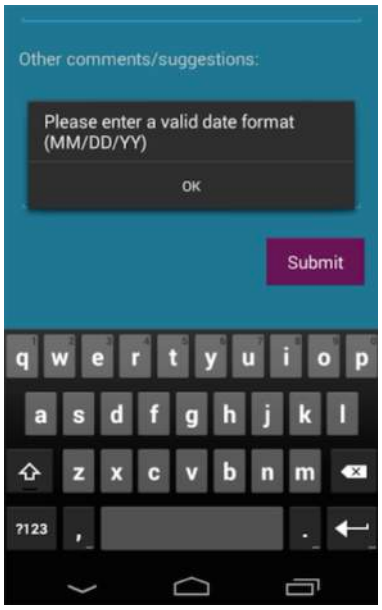
Please Wait....

a) **b)**

Figure 5. a) Fake network error during the realization of stage 3, b) Fake loading error that follows the fake network error.

The second frustrating event (FE2) occurred after subjects have completed all four stages and are asked to fill the survey form. After all the fields are filled and the submit button is pressed, an error dialog on formatting issues would popup (Fig. 6). The subject would dismiss the dialog, but only to find out that all previously entered data had been lost. Thus, the subject is forced to re-type all the answers again.

The specific error requests to include the date of birth in a valid format (MM/DD/YY). Had the subject already specified his/her birthdate in such format an error requesting a different format (such as MM/DD/YYYY) would have appeared nevertheless.



Please enter a valid date format (MM/DD/YY)

Figure 6. Frustrating event 2 (survey form). A date format error appears and all previously entered answers are deleted.

Summary and Conclusions

The experiment design follows a timeline that introduces each frustrating event at a known time t_f. This design allows for a controlled and systematic distinction of the baseline and frustration conditions for the purpose of ground-truth labeling that will be used in the analysis and classification of the collected data.

Two frustrating events are introduced during the experiment to foster a frustration classification approach that is context-independent and viable under naturalistic interaction with a smartphone (tapping, typing, scrolling down, etc.) in both landscape and portrait orientations.

CHAPTER 3

DATA COLLECTION

Participants & tasks

Twenty-one subjects were recruited from a senior-level computer science course to participate in the experiment described in chapter 2. Subjects were told that they were participating in a research experiment studying the effect of multitasking on cognitive loads and did not know the real goal of the experiment – determining whether data collected from a smartphone can seize user’s frustration – until the end of the session.

Subjects were provided with the same mobile phone (Nexus 4) to control for hardware variation. While subjects were performing the tasks, the device was recording sensor readings in the background: motion sensor readings such as the accelerometer and gyroscope, touch gestures, and facial videos captured using the front camera. The subjects were not made aware that these sensor recordings were taking place.

Subjects followed the experiment with the only constraint that they needed to hold the phone during the entire experiment and not have it placed it on any surface (e.g., a table). This constraint was imposed to guarantee that motion sensors could record all the hand movements of the subjects when interacting with their smartphone.

In order to incentivize competition and engagement, subjects were told that their final scores would be ranked among their fellow colleagues in class.

After subjects had completed the tasks, they informally reported how they felt about the tasks and whether they encountered any errors. All subjects except for subject #1 mentioned that they encountered some technical errors. Afterwards, it was revealed to the subjects the true intention of the experiment, and that the errors were “planted” to frustrate them. They all reported that they had believed the errors were genuine.

Manual face video analysis

Although the presented experiment had been designed to deliberately induce frustration, it is hard to determine whether the experiment succeeded to do so for each of the subjects. The data collected from the twenty-one subjects in the pilot study, despite the small number of subjects, reveals how self-reporting can be highly unreliable. For instance, subjects with identification numbers 9 and 19 reported not feeling frustrated or even bothered at all, but the front camera recordings clearly shows the opposite with an obvious expression of frustration. On the other hand, subject #8 reported a high level of frustration but not visible change in his facial expression can be perceived during any of the two frustrating events. Furthermore, subject #1 front camera video shows a frustrating facial expression during both frustrating events. However, he had forgotten about them after the experiment as he failed to report any of the errors when he was asked whether he had encountered any problem during the experiment.

Video recordings for each of the subjects were manually inspected to track down noticeable reactions of frustration. A clear physical indication of frustration was observed in 11 subjects during FE1 and 13 during FE2. A state of frustration cannot be associated with the rest of the subjects for several reasons: inexpressive facial expressions[1], bad illumination of recording[2], occlusion of most subject's face by subject's finger[3], face is out-of-view or too close to the camera[4] or total lost of video recording[5] due to technical problems. Sample images of expressive subjects can be seen in Fig. 7. Finger occlusion and partial out-of-view face samples are shown in Fig. 8.

Note that it has been assumed that a clear, strong frustrating reaction is an indicator of frustration, but that the lack of any facial expression does not exclude the possibility of a self-contained frustrated feeling (e.g., a person seated in a meeting is giving a very calmed and secure speech, but his/her right leg is shaking nervously under the table).

Table 2 indicates with a checked mark (✓) the subjects that have been annotated as becoming frustrated after the onset of FE1 or FE2. The question mark (?) indicates that a frustrating expression could not be identified due to any of the aforementioned reasons (numbered 1-5). Finally, an exclamation mark (⚠) indicates corrupted data. The corruption of the data occurred for different reasons: lack of smartphone memory storage to store the data[6], no sound played due to a programming error[7], and a subject interrupting the game to inform about the error in the middle of the experiment[8]. These data had to be discarded from the dataset, ending up with a dataset size of 17 subjects for

FE1 and 19 subjects for FE1, from which 11 and 13 subjects, respectively, have been labeled as becoming frustrated. The rest of the data is kept unlabeled.

	Did subject become frustrated?	
Subject ID	**FE1**	**FE2**
1	✓	✓
2	✓	✓
3	✓	✓
4	✓	✓
5	⚠^{6}	⚠^{6}
6	?^{5}	?^{5}
7	✓	⚠^{6}
8	?^{1}	?^{1}
9	✓	✓
10	?^{3}	✓
11	✓	✓
12	⚠^{7}	?^{1}
13	✓	✓
14	⚠^{8}	✓
15	⚠^{7}	?^{1}
16	✓	✓
17	?^{1}	?^{2}
18	?^{2}	✓
19	✓	✓
20	?^{4}	?^{4}
21	✓	✓

Table 2. Manually labeled subjects are indicated with a checked mark. Unlabeled subjects are designated with a question mark and numbered 1 – 5 indicating the reason for remaining unlabeled. Data that is lost or corrupted is shown with an exclamation mark and accompanied with their corresponding cause (numbered 6-8).

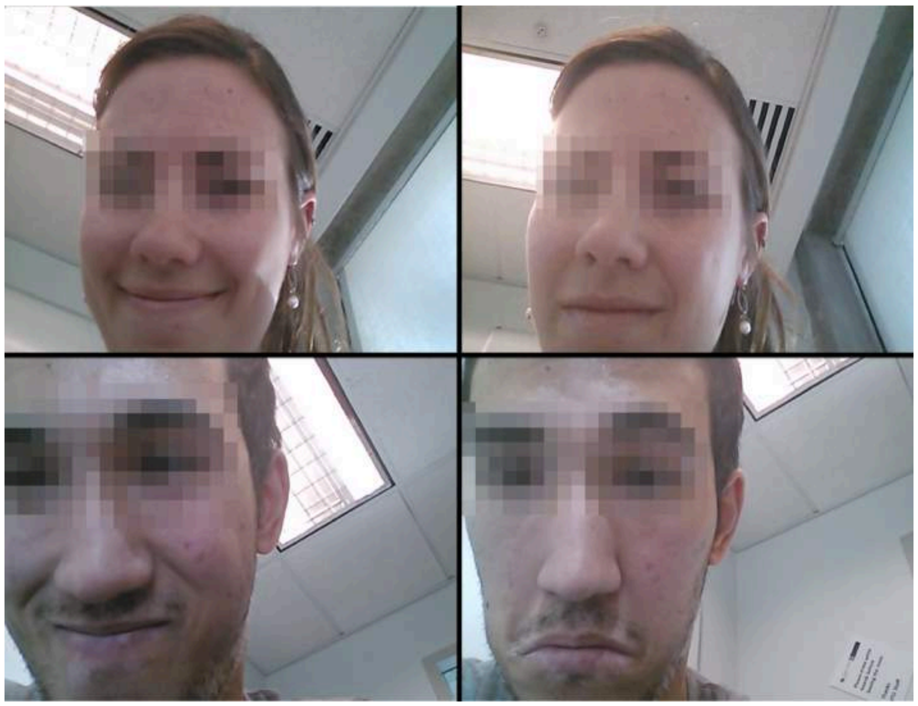


Figure 7. Sample faces showing different facial expressions.

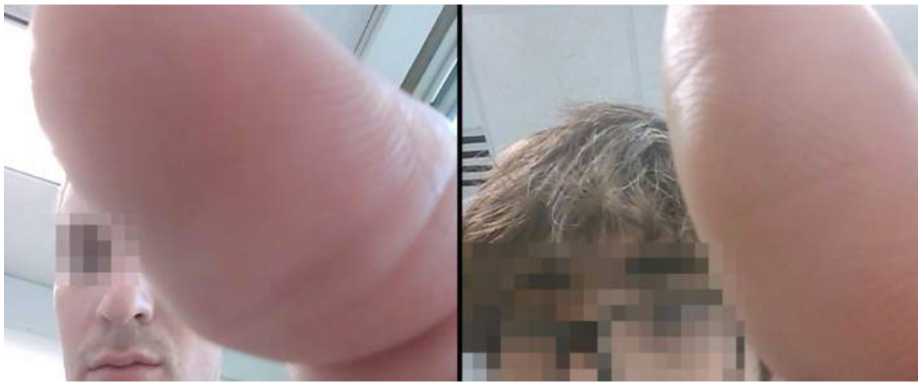


Figure 8. Finger occlusion and partial out-of-view face due to a close camera distance.

Summary and Conclusions

The review of the frontal camera videos while subjects performed the experiment has shown that subject self-reporting is inaccurate. For example, some subjects reported not becoming frustrated or even bothered by any of the frustrating events, whilst the reactions observed from the videos shows the opposite. Subject #1 had even forgotten about the errors after the experiment.

Reviewing the subjects' facial expressions and reactions during the experiment was also helpful to reaffirm that the frustrating events programmed in the experiment succeeded in frustrating many subjects. In particular, 11 and 13 subjects were labeled as frustrated subjects after FE1 and FE2, respectively.

Unfortunately, the dataset had to be reduced due to technical errors from 21 subjects to 17 and 19 for FE1 and FE2, respectively. The purpose of this data collection was to conduct a pilot study to determine to what extent data collected from a smartphone can seize user’s frustration. The results derived from this dataset will serve as proof of concept but a bigger dataset should be used to guarantee generalization over a bigger population.

CHAPTER 4

DATA PREPROCESSING AND FEATURE EXTRACTION

Smartphones recorded raw data in the background while subjects were undertaking the experiment. Collected data includes motion sensor readings (e.g. accelerometer and gyroscope readings), touch gestures and video recordings extracted from the front camera.

Whilst the videos extracted from the front camera have been used to corroborate the success of the experiment in the goal of inducing frustration, they were not further processed for the recognition of frustration using computer vision techniques due to the problems inherited from extracting video recordings from the front camera for the purpose of face detection [19] in addition to the issues that other researches have pointed out [9, 10].

The focus of the collected data for the purpose of data analysis and classification falls into two different modalities: touch gestures and motion sensor readings. The raw data from these modalities was preprocessed for feature extraction.

Touch gestures

A touch gesture is started when a finger touches the screen and it finishes when the finger goes back up, defining a trace that starts at position (x_{t_1}, y_{t_1}) and ends at position (x_{t_n}, y_{t_n}). At each timestamp $t_1, \dots, t_n$ raw touch information includes its coordinates, the pressure and size of the area covered on the screen, finger orientation, touch count (to handle multi-touch), and an event code (e.g. finger up, finger down, finger move, multi-touch).

The trace of a sample stroke that involves 12 events is shown in Fig. 9 and its raw stream of events is shown in Fig. 10.

Preprocessing

Individual touch gestures were identified from the data stream using the event codes; they started with an *action down* code and ended with an *action up* code. A touch gesture can range from 3 touch events (e.g. a very quick tap) to more than 50 events (e.g.

scrolling down or up). Features were extracted for each individual touch gesture, independently of its length or type.

Note that only touch gestures from playing the multitasking game were recorded. That is, the tap involved in pressing “OK” after the error dialog that appears in FE1 or any touch gesture that might have happened during the loading dialogs were discarded. Remember that the goal is to capture touch gestures of the same nature before and after a frustration event occurs. Hence, touch gestures that are directly related to the errors themselves are not considered for analysis and classification purposes.

Similarly, all touch gestures generated during the first 3 – 7 seconds (depending on the subject) that followed FE2 were also discarded. The reason is that every subject spent a few seconds scrolling all the form down and up again to confirm that, effectively, all the answers had been deleted after the fake date format error. Only the touch gestures that were effectuated after this particular scrolling action were kept.

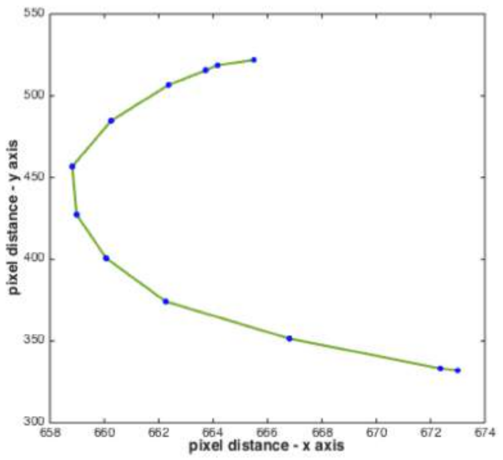
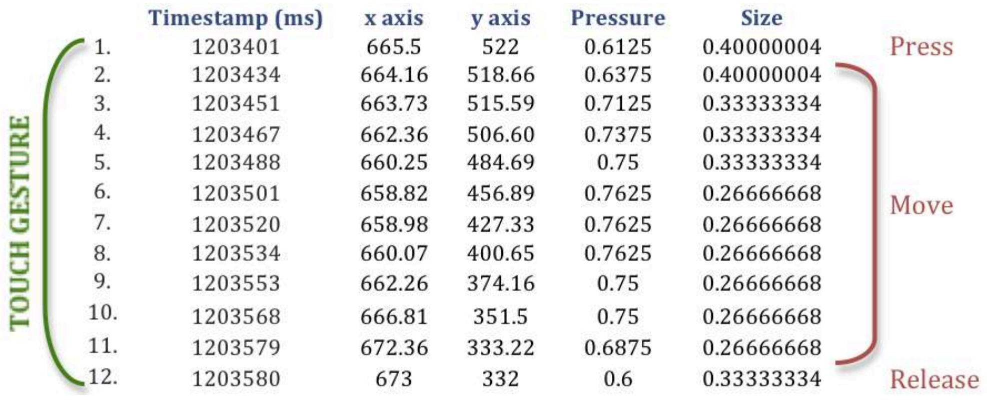


Figure 9. Visualization of a touch gesture.


TOUCH GESTURE
Press
Move
Release

	Timestamp (ms)	x axis	y axis	Pressure	Size
1.	1203401	665.5	522	0.6125	0.40000004
2.	1203434	664.16	518.66	0.6375	0.40000004
3.	1203451	663.73	515.59	0.7125	0.33333334
4.	1203467	662.36	506.60	0.7375	0.33333334
5.	1203488	660.25	484.69	0.75	0.33333334
6.	1203501	658.82	456.89	0.7625	0.26666668
7.	1203520	658.98	427.33	0.7625	0.26666668
8.	1203534	660.07	400.65	0.7625	0.26666668
9.	1203553	662.26	374.16	0.75	0.26666668
10.	1203568	666.81	351.5	0.75	0.26666668
11.	1203579	672.36	333.22	0.6875	0.26666668
12.	1203580	673	332	0.6	0.33333334

Figure 10. Stream of events representing a touch gesture.

Feature extraction

A set of 35 features adapted from [21] was extracted per each touch gesture. Some of these features include: touch duration, trace direction, trace length, velocity, acceleration, mid-stroke pressure, etc. A complete list of these features can be found in Table 3. More detailed information about these features can be found in [21].

Feature type	**Description**	**# features**
Distance	Distance in x-axis, distance in y-axis, Euclidean full distance, sum of pairwise distance (end-to-end line length)	4
Start, mid and end points	Mid-stroke pressure, Mid-stroke area covered, mid-stroke finger orientation, start and end pressure, start and end area covered	7
Time	Stroke duration	1
Velocity	Average velocity, Median velocity at first 3 points, median velocity at last 3 points	3
Acceleration	Median acceleration at first 3 points, median acceleration at last 3 points	2
Percentile	20%, 50% and 80%- percentile pairwise velocity 20%, 50% and 80%- percentile pairwise acceleration 20%, 50% and 80%- percentile deviation from end-to-end line	9
Circular distance (rad)	Angle traveled, end-to-end line direction, average pairwise angle traveled, flag direction (1-up, 2-down, 3-left, 4-right)	4
Other	Length of the major axis of an ellipse that describes the touch area, largest deviation from end-to-end line, ratio Euclidean distance and average pairwise angle traveled, change of finger orientation, touch count.	5

Table 3. Features extracted from a touch gesture.

Motion sensors

Motion sensors measure acceleration and rotational forces along the x-, y-, and z-axes. The coordinate system used in Android devices is defined relative to the device orientation as shown in Fig. 11. When a user holds the phone in portrait orientation, the x axis points to the right, the y axis points up and the z axis points toward the outside of the screen face [22].

During the experiment, data from two hardware-based motion sensors along the three physical axes was collected. The two motion sensors are:

- Accelerometer: measures the acceleration force applied to the smartphone. Note that the gravity force is included on all three axes.
- Gyroscope: measures the smartphone's rotation rate around the axes.

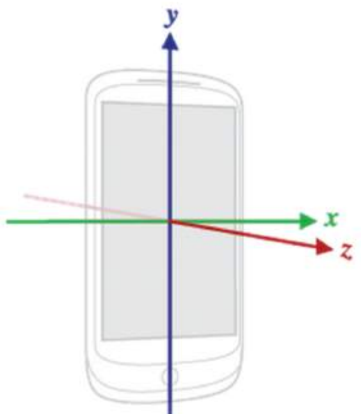


Figure 11. Coordinate system (relative to the device) used by the Android API [22].

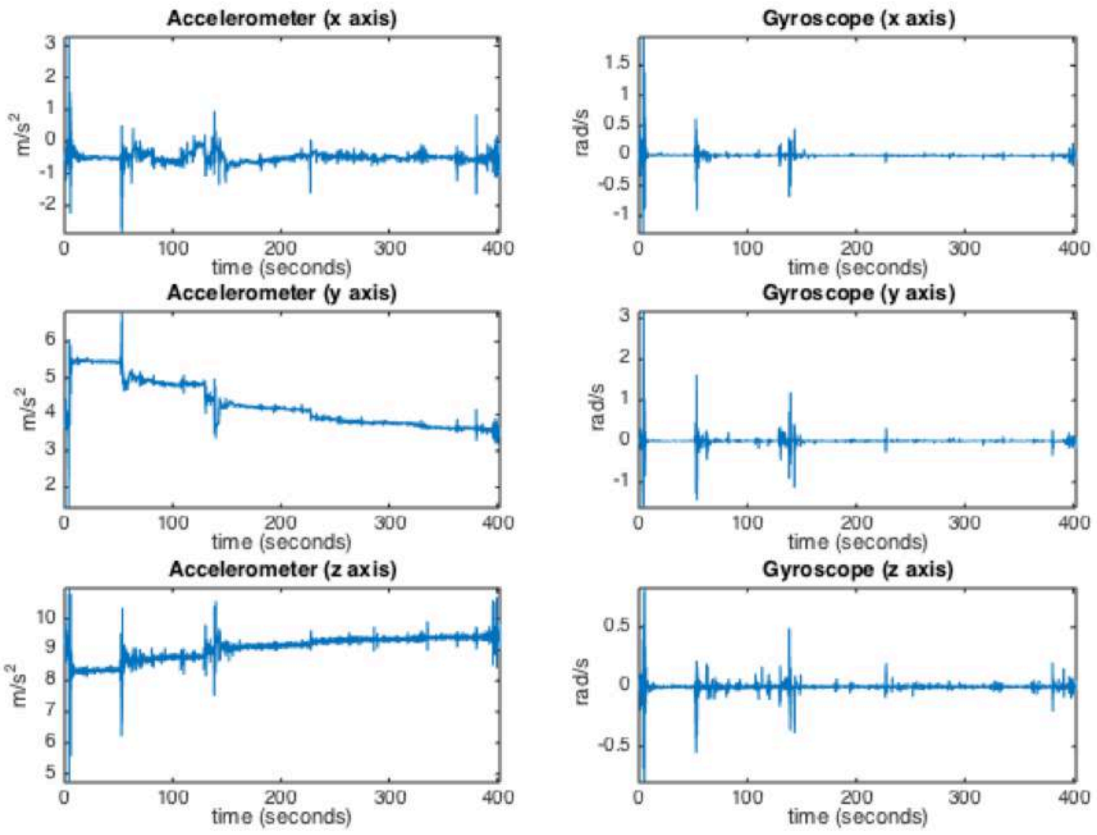


Figure 12. Fragment of accelerometer and gyroscope readings during the experiment.

Preprocessing

Sensor readings were preprocessed for noise reduction. Sensor noise can be observed when extracting motion sensor readings from a device being placed in a stationary position (e.g. placed on a table) as shown in Fig. 13. A median filter was applied to remove the presence of spikes and reduce the overall level of noise.

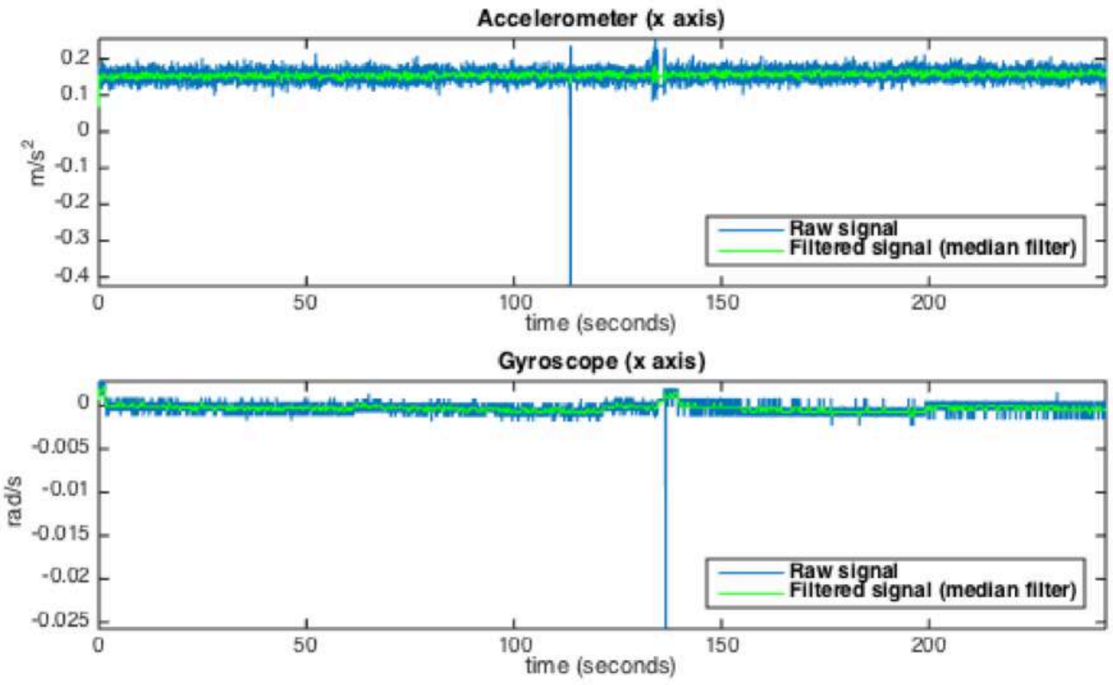


Figure 13. Accelerometer and gyroscope noise (x-axis only). A median filter removes the presence of spikes and reduces the overall level of noise.

Sensor events are triggered every time a new motion sensor reading is available. This occurs every 10 – 20 milliseconds at a non-uniform rate. In order to provide a uniform rate of 100 Hz, sensor values were interpolated using piecewise cubic hermite interpolating polynomial (PCHIP), which guarantees that no new extrema (minimum or maximum values) are introduced by the spline approximation of the data.

Finally, sensor signals were divided into 2-second windows with a 25% overlap between windows and feature extraction was performed for every window.

Feature extraction

Features were extracted at each 2-second window in the time, frequency and wavelet domains.

- **Time domain** features include features such as the mean, standard deviation, variance, minimum and maximum values or the energy of the signal. A more detailed description of the features can be seen in Table 4.

Feature type	Description	# features
Moments	Mean, absolute value of the mean, standard deviation, variance, skewness, kurtosis	6
Extremes	Min, max, absolute value of min and max	4
Energy	Energy of the 2-second window signal	1
Zero-crossing rate	Rate of sign-changes	1
Other statistics	Mode, sum of the absolute value of pairwise differences, inter-quartile range, histogram of z-scores (10 bins)	13

Table 4. Features extracted from accelerometer and gyroscope readings in the time domain.

- **Frequency domain** features require the transformation of the time signal into the frequency domain via a Discrete Fourier Transform (DFT). Features such as the energy, the centroid or the entropy of the signal in the frequency domain are extracted. Inspired by features used in audio recognition, the Mel-frequency cepstral coefficients (MFCCs) were also extracted. The full list of features in the frequency domain is shown in Table 5.

Feature type	Description	# features
Spectral energy	Energy of the signal' spectrum	1
Spectral centroid	"Center of mass" of the signal' spectrum	1
Spectral entropy	Entropy of the signal' spectrum	1
Spectral Roll-off	Amount of right skewness of the signal' spectrum	1
Harmonic	Harmonic of the signal	1
MFCCs	Coefficients that comprises a Mel-frequency cepstrum; a representation of the short-term spectrum	12

Table 5. Features extracted from accelerometer and gyroscope readings in the frequency domain.

- **Wavelet domain** features are extracted using the wavelet transform. Wavelets enclose a basis of functions used to decompose a signal by translation and scaling operations of a mother wavelet. Similarly, the Fourier transform can be seen as special case of a wavelet transform where the mother wavelet is $e^{-2\pi jt}$, and hence providing a frequency representation of the signal. Wavelets, however, provide a time-frequency representation of a signal. An in-depth reading on wavelet decomposition can be found in [23].

Features were extracted at each level of representation (for 5 levels) using the wavelet packet decomposition method described in [24] with a Haar wavelet (see Fig. 14), providing a total of 90 features per 2-second window signal.

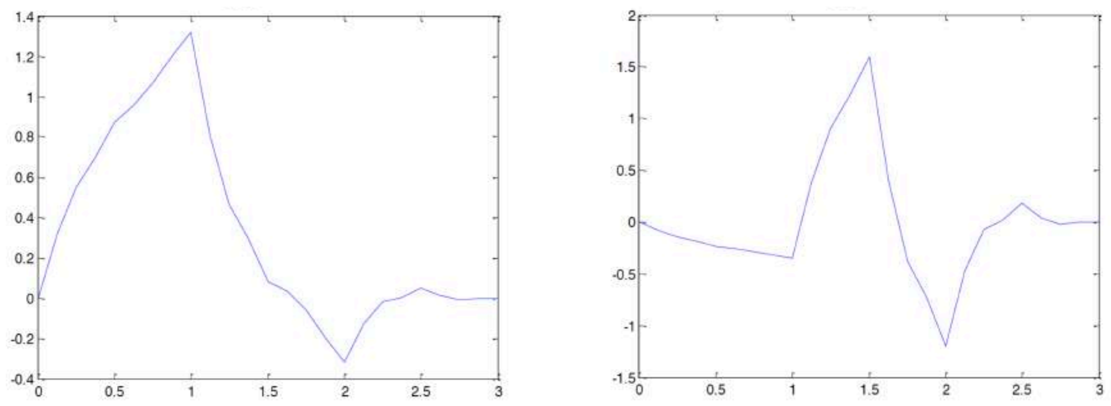


Figure 14. Haar scaling (left) and wavelet (right) functions.

The combination of accelerometer and gyroscope features along the three axes composes a feature vector of 792 features.

Summary and conclusions

Touch gestures were identified from the stream of touch events and a set of 35 features was extracted per touch. Raw motion sensor signals were preprocessed for noise reduction, and interpolated to match 100 Hz. About 800 features were extracted at every 2-second window signal. The choice of features has been inspired in other applications (e.g. speech recognition, authentication, etc.) but they have never been used before for the purpose of frustration detection.

CHAPTER 5

SUPERVISED CLASSIFCATION

In this chapter, the problem of frustration detection is treated as a binary (two-class) classification problem, where feature vectors collected prior to the onset of a FE are labeled as control, or non-frustrated ($\bar{F}$-class), and feature samples following the onset of a FE are labeled as pertaining to the *frustration* class (F-class). Note that two major assumptions have been made: 1) Subjects are either frustrated or non-frustrated; no other levels of frustration are considered, 2) frustration remains until the end of each stage.

Based on the analysis of subject's video recordings (Table 2), the dataset was divided into a supervised set ***S*** (corresponding to those subjects from which a frustrating reaction was observed), and an unsupervised set ***U*** with unknown labels.

Supervised classification is first carried out with the labeled set ***S***, formed by 11 subjects in FE1 and 13 subjects in FE2.

Overview of common classifiers in machine learning

The first goal of this pilot study was to confirm whether the extracted features specified in chapter 4 could serve to detect frustration using a supervised learning model.

Four well-known classifiers –logistic regression, neural networks, random forests and support vector machines– were tested to classify frustration using sensor and touch data streams independently. Refer to [25-29] for a more thorough mathematical description and formulation of these algorithms.

Logistic regression

Logistic regression is used to predict a binary class as a function of the input features using the sigmoid function. The sigmoid function is defined as $\sigma(z) = \frac{1}{1+ e^{-z}}$ and can take as input any value from $-\infty$ to $+\infty$ and models its output to a value between zero and one (see Fig. 15), which can be interpreted as the probability of an instance to be positive.

In the logistic function, z represents a linear function of the prediction variable of interest in terms of its features, where each feature, f_i, is multiplied by a weight, w_i, based on its prediction contribution. This step is usually known as linear regression, where $z = w_0 + w_1 f_1 + \ldots + w_n f_n$, and w_0 is a bias term introduced in the model.

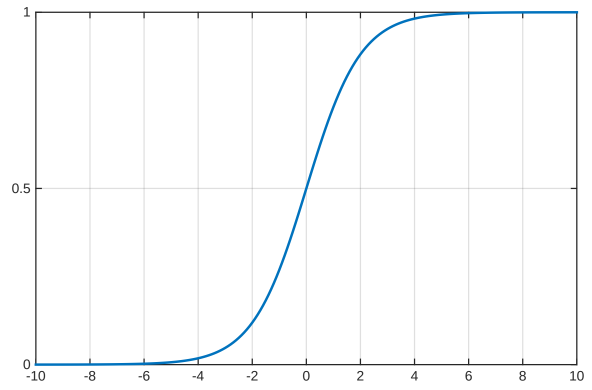


Figure 15. Logistic sigmoid function.

When learning the optimal weights, $\boldsymbol{w}^*$, a regularization term is usually added to z to apply a constraint on those weights. Regularization helps avoid overfitting and different types of regularization can be applied. For instance, adding an L1-regularization term $(+\lambda \sum_{i=1}^{n} |w_i|)$ promotes a sparse representation of the weights whilst an L2-regularization term $(+\lambda \sum_{i=1}^{n} w_i^2)$ penalizes large-valued weights.

Logistic regression is one of the most classic tools in discrete data analysis and statistics as it closely related to "exponential family" distributions, which are distributions that arise in many different contexts.

Neural networks (NN)

A neural network is a system inspired on the interconnection of neurons in human brains. It consists of interconnected inputs and outputs among several layers. Layers are made up of interconnected nodes that contain an activation function, which is usually non-linear (e.g. sigmoid function). The communication of data from the output neurons of a particular layer to the input neurons of the following layer through the activation functions on each node makes neural networks capable of approximating non-linear functions of their inputs.

A common learning technique (and the technique used in upcoming experiments) is backpropagation, where the weights are learned through a forward activation flow of outputs, and the backwards error propagation of newly adjusted weights.

The use of regularization techniques is common in neural networks to avoid overfitting. However, other modern techniques such as dropout also act in a regularization matter and have been shown to be more robust to noise [30]. The dropout technique consists of randomly "disconnecting" connections across the neural network architecture (see Fig. 16). The number of dropout connections is selected with a dropout rate parameter and these connections are randomly selected at each epoch during training.

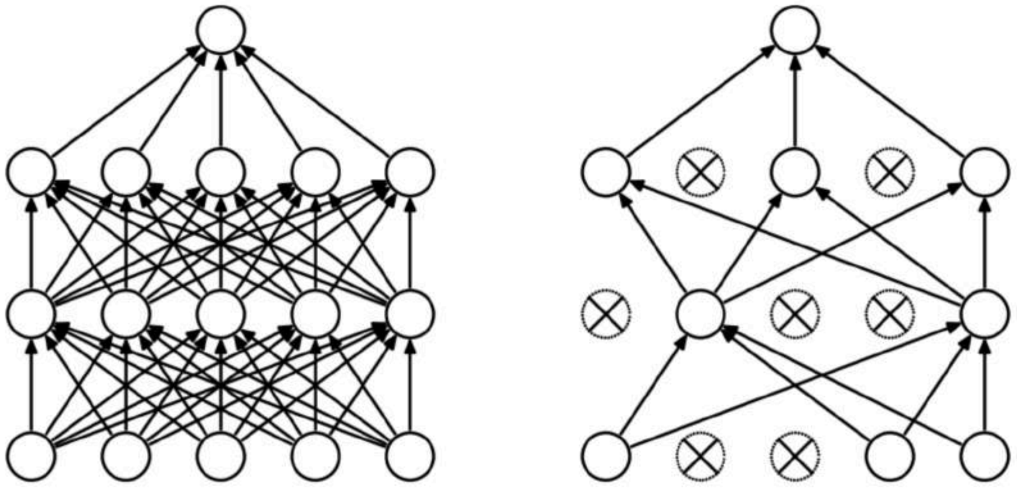


Figure 16. a) Standard neural network (all nodes are interconnected), b) Neural network after applying dropout [30].

Neural networks have become very popular due to their strong learning capabilities (usually referred to as "deep learning" when many hidden layers are used). However, they usually require a lot of learning time and parameter optimization to obtain high prediction results.

Random forest

A random is an ensemble of decision tree classifiers, which are trained on random sub-samples of the dataset. A decision tree represents the data in a tree-like structure, where leaves represent class labels and branches represent a feature or group of features that lead to a class label. The "forest" chooses the classification having the most votes, where each tree gives a classification and "votes" for a particular class. By averaging multiple decision trees, the predictive accuracy is remarkably improved and is more robust to over-fitting.

Random forests are very popular due to their very few parameters to optimize for, providing good accuracies with just the default parameter settings in many prediction problems.

Support vector machines (SVM)

An SVM is a binary classification technique that constructs a linear hyperplane (or a set of hyperplanes in a high-dimensional space) that best separate two different classes by maximizing the margin between the data points that delimit each of the two classes (and these data points used to define the widest gap between the two classes are called support vectors).

A simple 2-dimensional example is shown in Fig. 17, where the line shows the best division found by the SVM between the two classes. However, some problems do not have a linearly separable hyperplane between the two classes. In those cases, non-linear classification can be performed with an SVM via the “kernel trick”, which maps the data into a high-dimensional feature space where an optimal hyperplane can be found to better separate the data. Common kernels are the polynomial and the radial basis function (Gaussian), which can be efficiently applied to the original data with the dot product operator.

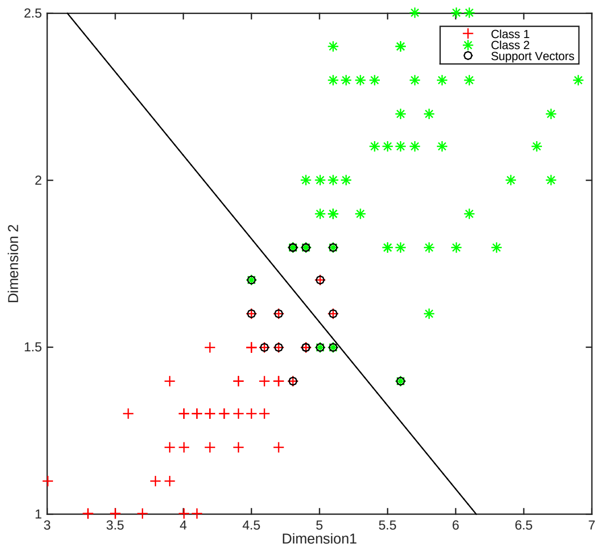


Figure 17. SVM example. Support vectors define the greatest hyperplane that best separates the two classes.

Feature reduction with Principal Component Analysis

Principal Component Analysis (PCA) is a popular linear, unsupervised technique often used for dimensionality reduction. It consists of decomposing the covariance matrix $C = XX^T$ (where X is the feature matrix) into $C = UVU^T$. The matrix U contains the eigenvectors and V is a diagonal matrix containing the eigenvalues corresponding to each eigenvectors. Eigenvectors are orthogonal and point in the direction of greatest variance, while the eigenvalues are a scale factor for each eigenvector. Fig. 18 shows a simple 2-dimensional example of a multivariate Gaussian distribution, where z_1 and z_2 are the eigenvectors of the covariance matrix *C*.

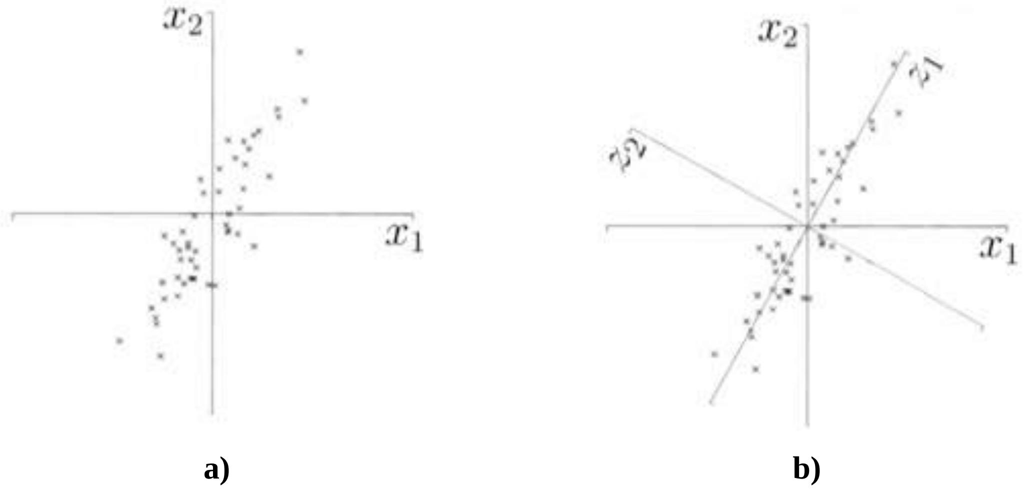


Figure 18. a) Original data, b) PCA decides to project onto the plane z_1 (1st principal component) and z_2.

Small eigenvalues indicate little or no variation of the data. Dimensionality reduction can therefore be achieved by rejecting the M least relevant eigenvectors, which are often noise or redundant data.

Many other dimensionality reduction techniques exist but PCA has been widely used to evaluate the benefits of feature reduction under different classification techniques.

Generalization and Cross-validation

In a prediction problem, a model is usually trained on an ideally large training dataset. Such model is then tested on a testing dataset that is independent from the training dataset. A classifier that gives good performance on the training set might not generalize well on the testing dataset due to overfitting of the training data.

Cross-validation is used to assess the generalization of a classification model to an independent data set, as well as to optimize the selection of a model and its model parameters and to avoid overfitting.

Cross-validation consists of partitioning the dataset into complementary subsets, and training the model using one subset (the training set) and validating the performance on another subset (the testing or validation set). This is done multiple times using different partitions each time, and the performance results are averaged over the cross-validation runs.

There are different cross-validation set-ups that split the data in different ways. One extreme way to partition the data is to select one single data point for test and the remaining data for training. When done iteratively for each data point, this is called leave-one-out cross-validation.

Leave-one-out cross-validation is an exhaustive method for cross-validation but it is not always suitable when the data points are not completely independent. For instance, the dataset presented in chapter 3 contains several instances of touch gestures and motion sensor readings per each of the available subjects. Instances from the same subject are not independent, whether they come from the control or frustration conditions. For this reason, leave-one-subject-out cross-validation (LOSOCV) has been used in every classification setup to guarantee that no within-subject characteristic traits are retained and used by the classifier, while allowing intra-subject behavioral traits to be extracted for the purpose of frustration detection and guarantee generalization over subjects. During LOSOCV, all data points corresponding to one subject are used for testing while the remaining subjects are used for training.

ROC curves

Binary classification performance is commonly illustrated with a receiver operator characteristic (ROC) curve. It shows the trade-off between the true positive rate and the false positive rate at various threshold settings. The true positive rate measures the proportion of positives –touch gestures and motion sensor readings during frustration conditions– which are correctly classified to the F-class. On the other hand, the false positive rate indicates the rate at which instances during control conditions are incorrectly assigned to the F-class.

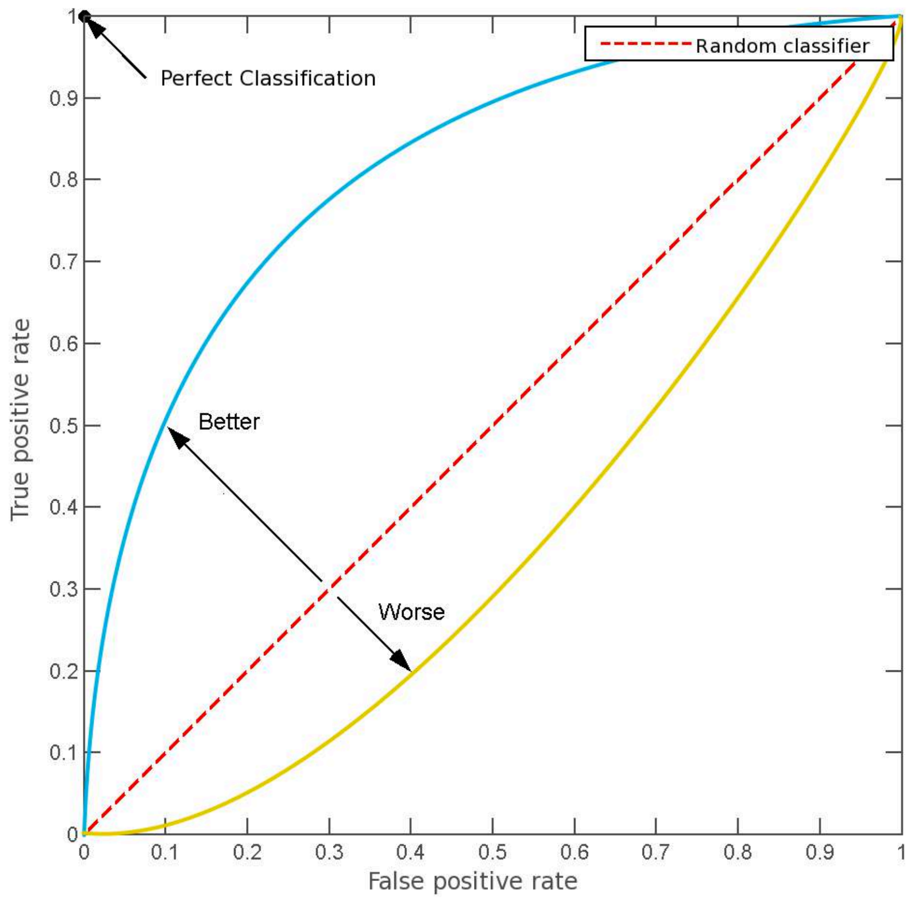


Figure 19. The ROC space. A random classifier has an AUROC of 0.5, while better classifiers will present a curve with an AUROC > 0.5.

The area under the ROC curve (AUROC) is used for a single-metric performance comparison of the classifiers tested. A perfect classifier has an AUROC of 1 (i.e. a true positive rate of 1 and a false positive rate of 0), whilst a random classifier presents an AUROC of 0.5, and anything below 0.5 performs worst than a random guess. Fig. 19 shows an example of different ROC curves.

ROC curves and AUROC values will be used in the following sections to compare performance of the different frustration classification systems presented.

Classification performance with touch gestures

In this section, four popular classification methods were tested to predict frustration using touch gesture features specified in chapter 4. The classification methods used are: logistic regression, neural networks, random forests and support vector machines.

The classification performance obtained from each classifier using LOSOCV is shown in Table 6. Each classifier was trained for FE1 and FE2, using the full feature set of 35 touch features or a projected version of those features using PCA. In this first setup, each touch gesture is classified independently and assigned one class ($\bar{F}$ or F). That is, no timing information has been used to assign touch gestures that occurred close in time.

	FE1		**FE2**	
	Full feature set (35)	**PCA features (17)**	**Full feature set (35)**	**PCA features (17)**
Logistic regression	**0.6615**	0.6409	**0.6444**	0.6368
NN	0.6388	0.6339	0.6225	0.6362
Random Forest	0.6311	0.6038	0.6174	0.5992
SVM	0.6597	0.6610	0.6425	0.6261

Table 6. Touch gesture classifier: AUROC performance comparison for four different classifiers. The best AUROC is shown in bold for each frustrating event.

The use of PCA reduced the dimensionality of the feature space from 35 features to 17 when keeping 99% of the variance. However, no benefits were obtained when using PCA features instead of the full set of features.

Via cross-validation, some parameters were optimized to obtain higher classification accuracies. For instance, logistic regression provided the best performance when using L1 regularization with a parameter of $\lambda = 2$. Neural networks seemed to provide a better performance when using two hidden layers and a dropout rate of 50%. A linear SVM was chosen as no significant improvement was found when using a Gaussian or polynomial kernel with the same set of features.

All of the classifiers performed above chance with either a full or reduced feature set and frustrating event. Logistic regression with L1 regularization was selected as it provided the best results among the classifiers used, with an AUROC of 0.6615 and 0.6644 for FE1 and FE2, respectively, on the supervised set ***S***.

L1-regularization encourages a sparse solution by setting the weights that correspond to the least predictive features to zero. Non-zero weights points to the features that contributed to the logistic regression prediction, and can be seen as a built-in feature selection step inside the classifier. Identifying which features contributed the most in the predictions is important to get a better understanding of what could be indicating user frustration. Fig. 20 shows the box plot of the features with non-zero weights used by logistic regression. The features have been separated into control and frustration conditions to identify any statistical difference.

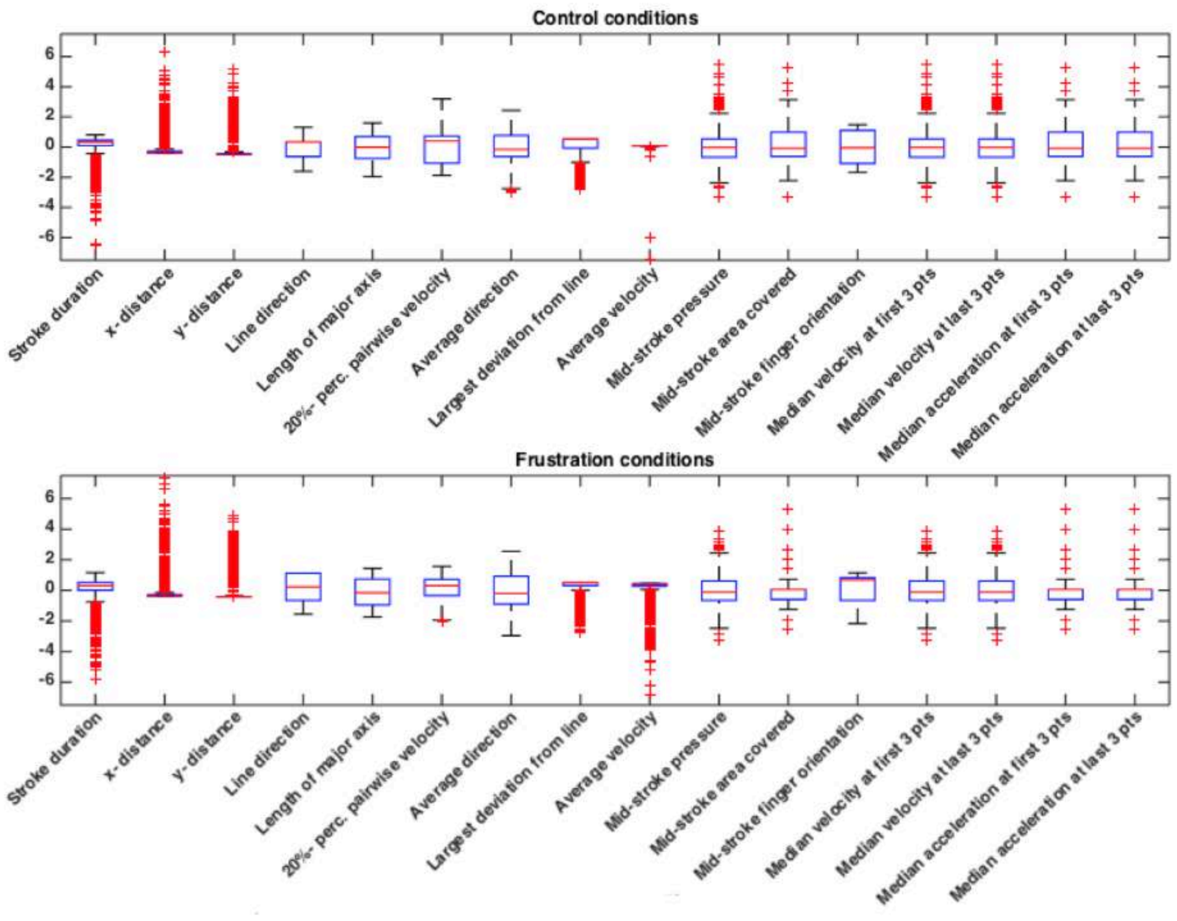


Figure 20. Box plots of the non-zero weighted touch gesture features (as obtained with logistic regression) under control and frustration conditions.

Classification performance with motion sensors

Similarly to the classification setups applied to touch gestures, Table 7 compares the classification performance in terms of AUROC with four classifiers. Again, these classifiers are logistic regression, neural networks, random forests and SVMs.

Here, motion sensor features extracted from each 2-second window are classified to pertain to the $\bar{F}$- or F-class using LOSOCV. At this stage, no timing information has been taken into account and windows close to each other are classified independently.

Logistic regression with L1-regularization and regularization parameter $\lambda = 10$ did not provide as good results with motion sensor data as it did with touch gestures. The best performance was obtained using a random forest of 100 trees, with an averaged AUROC of 0.7020 and 0.6383 for FE1 and FE2, respectively. Both a linear SVM and a neural network with two hidden layers (with dropout rate of 0.5) still performed above chance but did not outperform the random forest.

Fig. 21 shows the out-of-bag classification error in terms of the number of trees grown on the selected random forest. Out-of-bag classification error is computed with the data points that are out-of-bag (sub-sample used to grow a tree) and compared with the true labels at run-time as new trees are grown in the forest. A random forest with 100 trees was chosen, as it can be seen from the figure that adding more trees does not improve the performance while increasing the computing time.

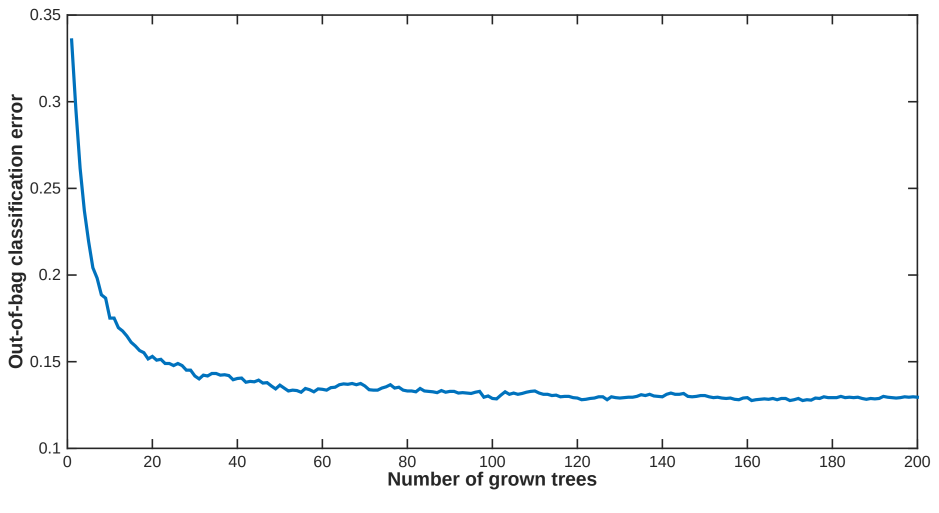


Figure 21. Out-of-bag classification error over the number of grown classification trees.

The use of PCA to reduce the dimensionality of the feature space, which reduced the number of features from 792 to 103 when keeping 99% of the variance, improved the performance of logistic regression in both frustrating scenarios. However, the use of PCA features caused a significant drop in the performance of random forests, and therefore the full feature set was finally used.

	FE1		**FE2**	
	Full feature set (792)	**PCA features (103)**	**Full feature set (792)**	**PCA features (103)**
Logistic regression	0.5531	0.5677	0.5882	0.6120
NN	0.6171	0.6274	0.5439	0.5379
Random Forest	**0.7020**	0.6296	**0.6383**	0.5331
SVM	0.6102	0.5869	0.604	0.6302

Table 7. Motion sensor classifier: AUROC performance comparison for four different classifiers. The best AUROC is shown in bold for each frustrating event.

Feature-level vs. decision-level fusion

The results shown in Table 7 were obtained using motion sensor features extracted in the time, frequency and wavelet domain that had been combined in a feature-level manner (see Fig. 22). That is, all types of features were concatenated and fed into the classifier as a 792-dimensional feature vector per 2-second window. The three types of features could also be combined in a decision-level manner, which fuses the posterior probabilities for any given 2-second window obtained from an ensemble of random forests, each trained on a different feature type.

Two decision-level fusion rules were tested; a linear weighted fusion and a SVM fusion. The linear weighted fusion rule applies a weighted average to the outputs of each independent random forest classifier, where the weights of each classifier have been set to their independent performance (using the AUROC metric) and normalized to sum 1. The SVM fusion rule (Fig. 23), on the other hand, consists of training a linear SVM to discriminate between the output classes F and $\bar{F}$ using the posterior probability values of each individual random forest.

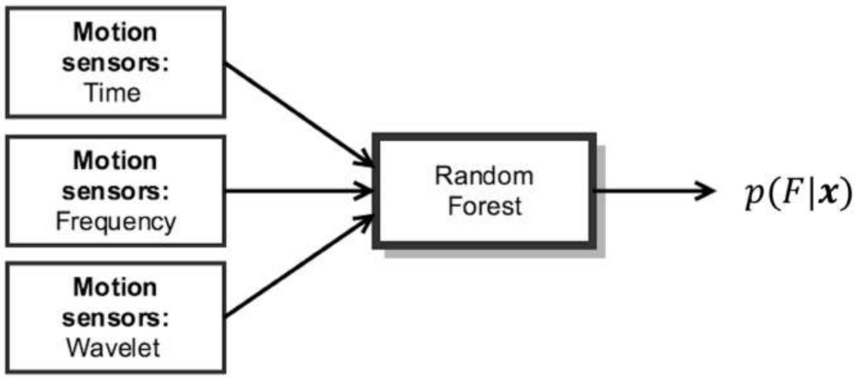


Figure 22. Feature-level fusion. Motion sensor features extracted in the time, frequency and wavelet domain are concatenated in the same feature vector *x*.

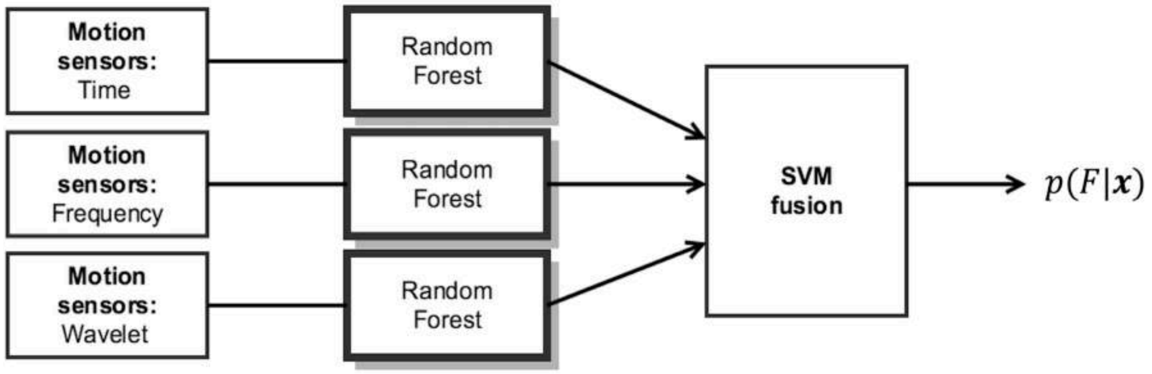


Figure 23. Decision-level fusion using SVM rule. Each feature type is classified independently using a random forest. Then, SVM fusion is applied.

Table 8 shows the individual contributions of each motion sensor feature group (time, frequency and wavelet). Time features present the best performance among the three modalities on both frustrating events, and are followed by the wavelet features. However, combining the three groups of features in a feature-level manner outperforms the use of any of the individual groups. This suggests that each of the feature groups contributes to the prediction of frustration. There is not a significant difference between the feature-level fusion and the linear weighted fusion. However, the last decision-level fusion method, the SVM fusion rule, outperforms the feature-level fusion in both frustrating events, with an AUROC of 0.7117 and 0.6511.

Modality fusion

Previous experiments were carried out with sensor and touch data independently, obtaining performance results above chance in every case. However, the combination of the two streams can increase robustness of the presented frustration detection system.

Sensor features are extracted every two seconds (using a 2-second window with 25% overlap) whilst touch gesture features are extracted whenever the user touches the screen. It is necessary to provide synchronization between the two streams to perform a decision-level fusion. This can be achieved by integrating the posterior probabilities over time. A 10-second window with a 50% overlap between windows was selected, which generates an averaged probability score for each modality every 5 seconds. Averaged probabilities are then fed into an SVM to combine the two modalities and generate a final score of frustration at each time window. If no touch has been effectuated during a given 10-second window, then no fusion is effectuated and only the averaged probability score resulting from the motion sensors modality is considered.

An overview of the frustration detection system is presented in Fig. 24. Each individual modality is classified using a separate random forest, and SVM fusion is used to provide a unique prediction. The first SVM fusion is trained on the different motion sensors feature types. The second SVM fusion combines the two modalities (motion sensors and touch gestures) after an integration of the individual probabilities along a 10-second window.

Integrating posterior probabilities over time not only offered a synchronized decision-level fusion but also equipped the touch gesture and motion sensor classifiers with robustness towards noisy predictions. The touch gesture predictor increased its AUROC measures by 14.53% and 5.99% for FE1 and FE2 respectively. In the case of the fused motion classifiers, on the other hand, the integration over a 10-second window increased the performance by 5.90% and 2.76%.

The employment of modality SVM fusion over the two synchronized data streams provides a remarkable boost to the overall performance as can be seen in Table 9, with final AUROC values of 0.8205 and 0.7734 for FE1 and FE2, respectively. Fig. 25 and Fig. 26 show the ROC curves for touch gestures and motion sensor classifiers before and after integration, and the final classification performance when combining the two modalities for each of the frustrating scenarios.

	FE1	FE2
Time features	0.6958	0.6259
Frequency features	0.6675	0.5879
Wavelet features	0.6803	0.6184
Feature-level fusion	0.7020	0.6383
Linear weighted fusion	0.6974	0.6371
SVM fusion	**0.7117**	**0.6511**

Table 8. Performance classification comparison (feature-level vs. decision-level fusion).

	FE1	FE2
Motion sensor classifier (window integration)	0.7537	0.6691
Touch gesture classifier (window integration)	0.7576	0.6830
Modality SVM fusion	**0.8205**	**0.7734**

Table 9. AUROC values for motion sensor and touch gestures classifiers after a 10-second window integration and the modality SVM fusion.

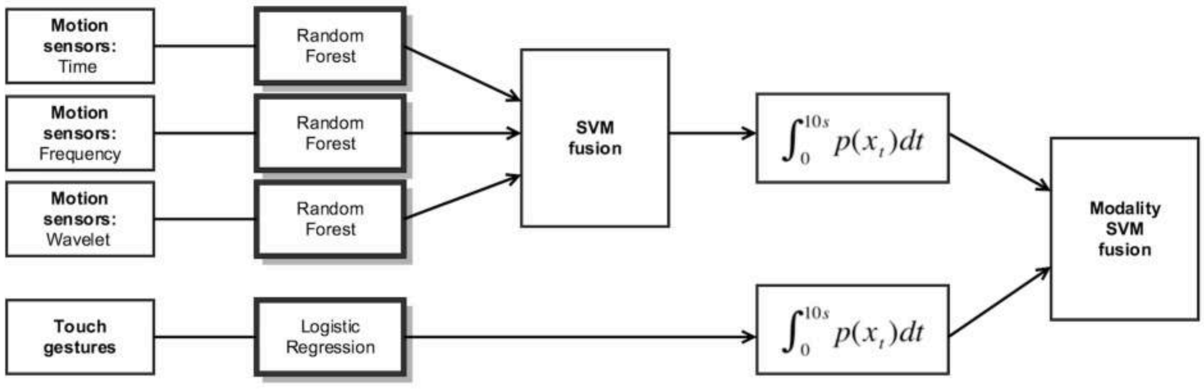


Figure 24. Frustration detection system overview. Features extracted from motion sensors are divided by domain (time, frequency and wavelet) and fed into an ensemble of random forests. Predictions are then fused using a SVM fusion rule. Touch gesture feature vectors are classified using logistic regression. Then, probabilities of frustration of each of the modalities are integrated over a 10-second window. Finally, prediction scores from both modalities are fused with SVM fusion.

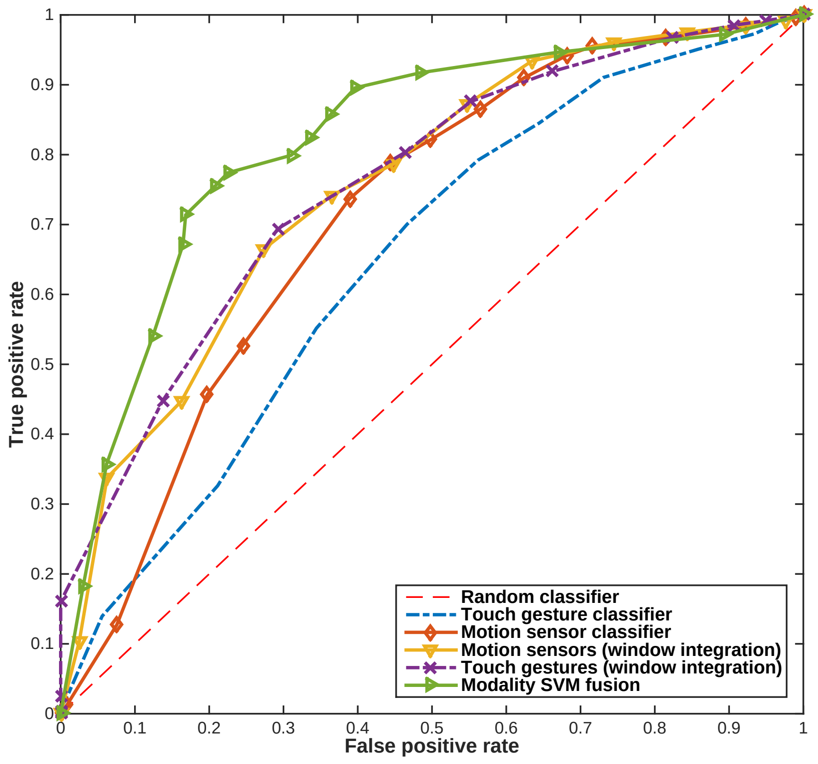


Figure 25. ROC curves during FE1 for single modality and modality SVM fusion classifiers, averaged after LOSOCV on supervised set *S*.

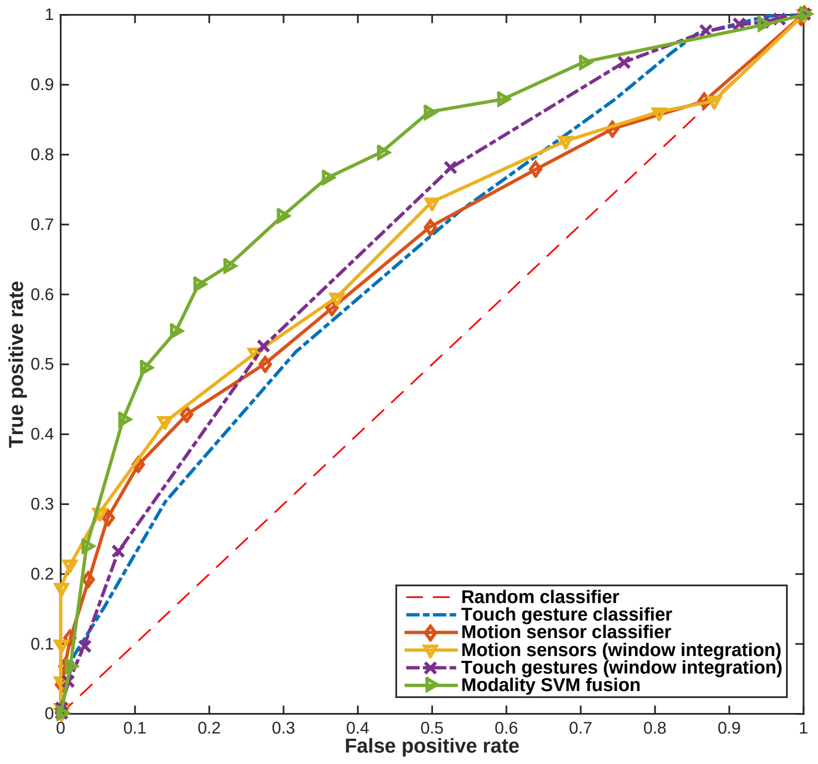


Figure 26. ROC curves during FE2 for single modality and modality SVM fusion classifiers, averaged after LOSOCV on supervised set *S*.

Summary and conclusions

The results obtained show that sensor data and touch gestures extracted from a smartphone can potentially determine frustration while users are naturally using a smartphone. Even when only one modality is used –either motion sensors or touch gestures– the classification results yield accuracies above chance. Reasonable performance has been obtained when the input features are integrated over a window of time, providing more robustness over the inherent noise in the measurements and the high variance of the data.

A remarkable boost on the performance is obtained when the two modalities are combined using a SVM fusion rule, leading to 0.8205 and 0.7734 AUROC for FE1 and FE2, respectively.

CHAPTER 6

HANDLING UNLABELED DATA

A system to predict frustration was implemented in chapter 5 using the supervised set ***S***, which was manually labeled based on the analysis of front camera recordings. Another subset of data could not be labeled for several reasons explained in chapter 3. This last subset of data is referred to as the unsupervised set ***U***, given that the true labels are not known.

Semi-supervised learning can be exploited when both supervised and unsupervised data are available, where supervised data is often used to label the unsupervised set.

In this chapter, semi-supervised learning is explored in the collected dataset to attempt to label the data from the unsupervised set ***U*** to learn which subjects became frustrated after a frustrating event.

Dimensionality reduction and data visualization

Clustering is a very common task in unsupervised and semi-supervised learning. It consists of separating the data into different groups (clusters) based on their similarity. It is based on the smoothness assumption, which states that data points that are close to each other are more likely to pertain to the same class.

One of the most popular clustering methods is *k*-means, which partitions the data into *k* clusters that "best" separates the data following an assumption of spherical clusters where a cluster mean (or centroid) corresponds to the cluster center after several iterations. Fig. 27 shows the output of *k*-means on toy data when $k = 2$. Two separable clusters have been identified and data points have been assigned to the closest cluster.

K-means did not provide a separability of the F and $\bar{F}$ classes higher than chance in either the touch gesture or motion sensor feature space. It is usually hard to analyze and understand high-dimensional data, as it is rather difficult to visualize.

Dimensionality reduction techniques can be applied for visualization purposes when the available data is further reduced to two or three dimensions. For instance, the 1st and 2nd principal components (which account for the highest variation of the data) have been plotted in Fig. 28 (top). It can be seen from the figure that instances from the F and $\bar{F}$-class are not easily separable into two well-defined clusters using the two principal components.

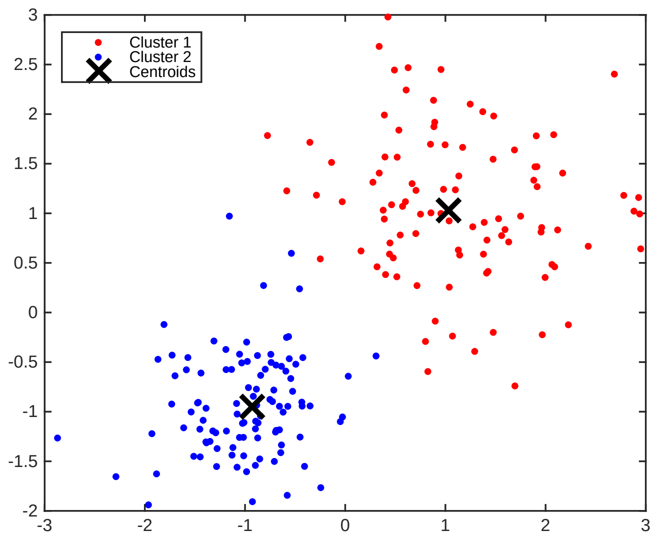


Figure 27. Cluster assignments and centroids (*k*-means).

With the hope that a non-linear model would capture the multi-modal aspects and non-linear factors of variation in the data, the use of deep autoencoders for dimensionality reduction was explored. A deep autoencoder has a deep neural network architecture that sets the target values to be equal to the inputs and optimizes the weights to minimize the error between the input and the output of the deep autoencoder (see Fig. 29). Hence, the deep architecture is learning a compressed representation of the data through the hidden layers. Deep autoencoders can learn interesting features in images and speech signals, and they provide a better initialization and faster training when used for deep learning pre-training [31].

Two deep autoencoder architectures with three hidden layers were applied on both touch gesture and motion sensor data. The last hidden layer (with 2 hidden units) was extracted and plotted in Fig. 28 (bottom). Although the variance of the data has been reduced, instances coming from control and frustration conditions are still not easily separable into two clusters.

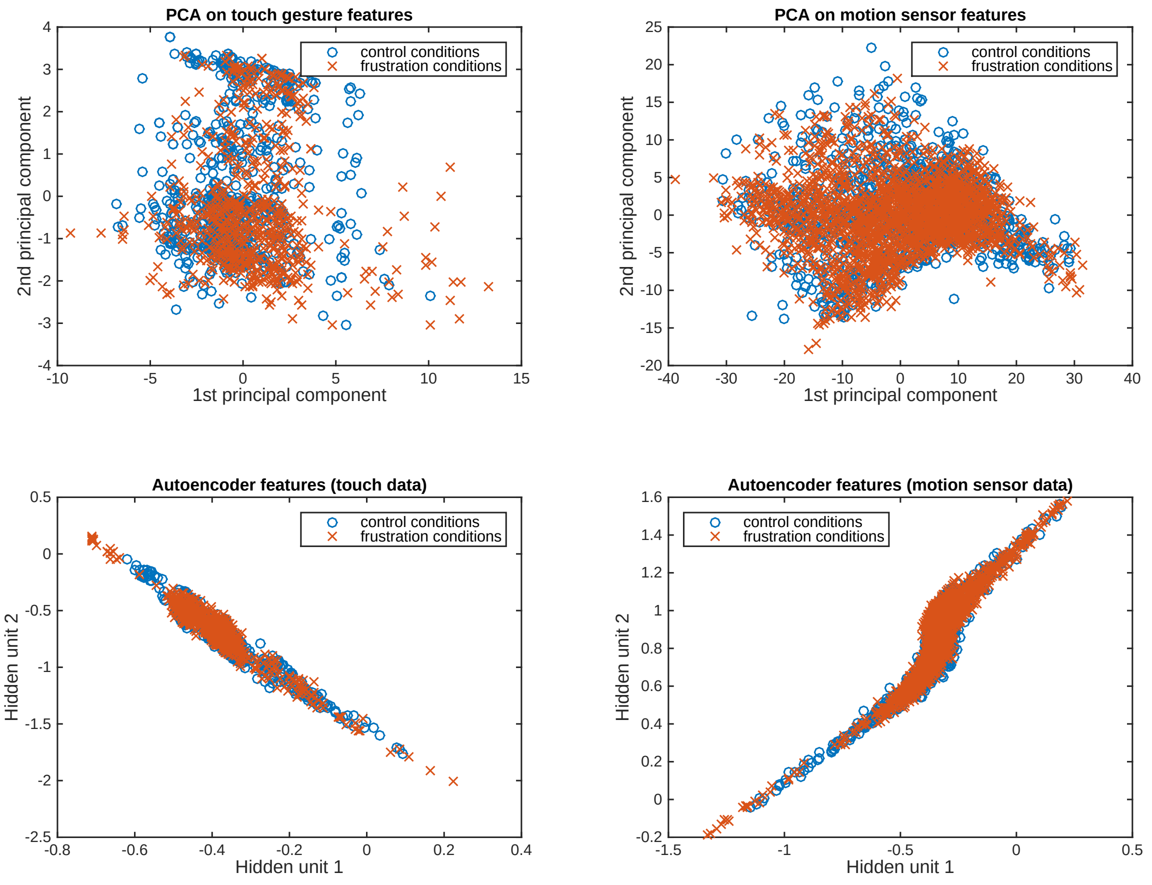


Figure 28. Visualization of data points in control and frustration conditions using dimensionality reduction: PCA (up) and deep autoencoder (down).

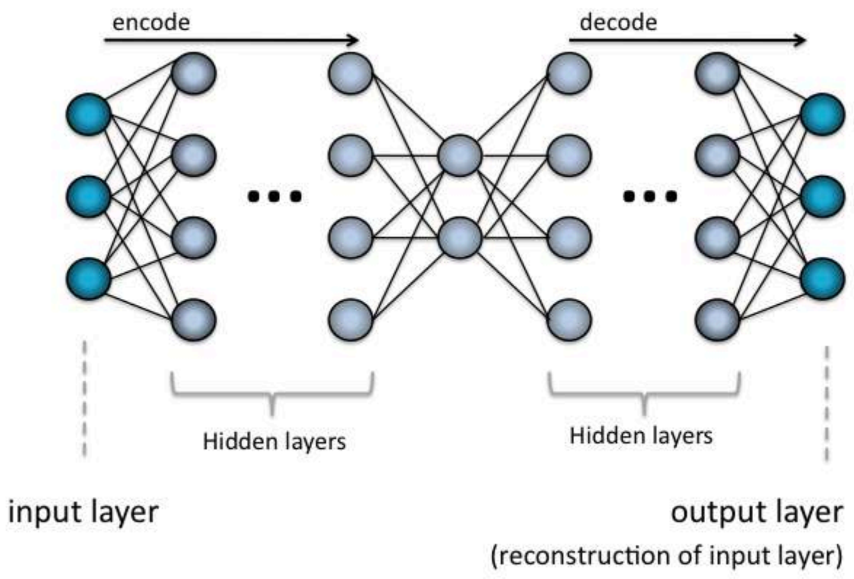


Figure 29. Deep autoencoder architecture.

Subject self-labeling

Although clustering algorithms were not the best fit to this dataset, a classification system with reasonable performance had been implemented in chapter 5. In this section, the system depicted in Fig. 24 is used to classify data from the unsupervised set $\boldsymbol{U}$ and assign the labels attached to each subject's data from $\boldsymbol{U}$. To do so, a semi-supervised algorithm based on self-training is used. This algorithm is referred to as subject self-labeling since individual data points are not considered independently but all data points pertaining to the same subject are grouped together.

Self-training is a heuristic approach that uses a supervised learning algorithm trained on the supervised set to label the unsupervised set with the expectation that using a higher volume of data will build a better classifier. Labels from the unlabeled set are iteratively added to the labeled set based on the confidence of each classified data point.

A special case of self-training is co-training, where the feature set is divided into two different classifiers. Co-training specially fits the problem of this thesis as two independent classifiers (one for touch gestures and one for motion sensors) have previously been implemented using the available labeled data. However, the complete system fuses both classifier outcomes with modality SVM fusion and therefore the implemented semi-supervised technique fits the definition of self-training better.

A correlation measure that uses information about the designed experiment has been used to identify which subjects were more likely to have become frustrated after a frustrating event. Specifically, Pearson's correlation is computed between the posterior probabilities after the modality SVM fusion and the ideal probability output of a frustrated user (i.e. a zero probability of frustration from the start of a particular stage and t_f, and a probability of frustration of 1 after t_f and until the end of the stage). Fig. 30 shows the probability of frustration as a function of time. Each instance is correlated with the ideal expected output represented as a step function in red. Pearson's correlation outputs values from -1 (i.e. inverse correlation) to +1 (i.e. perfect correlation).

Subject self-labeling applies the defined correlation measure. A threshold is needed to determine which subjects in $\boldsymbol{U}$ can be assumed to have become frustrated. It is not clear which value the threshold should take, as it is hard to analyze the performance of this semi-supervised approach. In the presented experiment, a hard-coded threshold of 0.5 has been chosen.

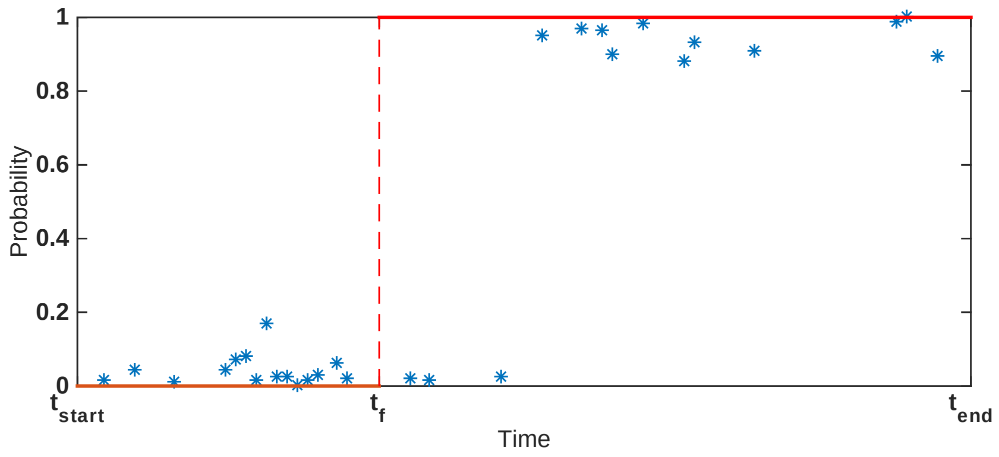


Figure 30. Frustration probability values during FE2 (subject #3).

The pseudo-code of the subject self-labeling algorithm applied is shown as follows:

1. Train frustration classification system (Fig. 24) with set ***S***.
2. Obtain frustration probabilities for ***U***.
3. Select subject s_i with the highest correlation value (with its ideal step signal).
4. If correlation is bigger than the threshold, then add subject s_i to ***S***.
5. Repeat 1- 4 until no more subjects can be added to ***S***.

After completion of subject self-labeling, the set ***S*** has turned into a new set ***S****, which contains the true labels from the supervised set in addition to the new self-learned labels from the unsupervised set.

Fig. 31 and 32 show the correlation values of each subject before and after subject self-labeling. Subjects pertaining to ***S*** are represented in a dark green shade and subjects from ***U*** are depicted in a dark purple color. After subject self-labeling (right column bars), subjects in ***U*** have been labeled as *F* if their Pearson's correlation value became equal or higher than 0.5. Otherwise, subjects would remain unlabeled, presumably because those subjects did not become frustrated after the corresponding FE.

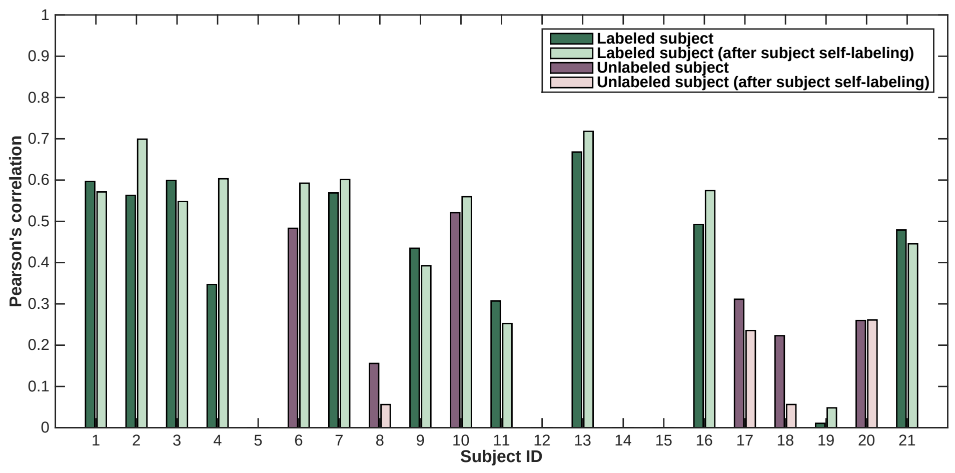


Figure 31. Pearson's correlation and labeling before and after subject self-labeling (FE1).

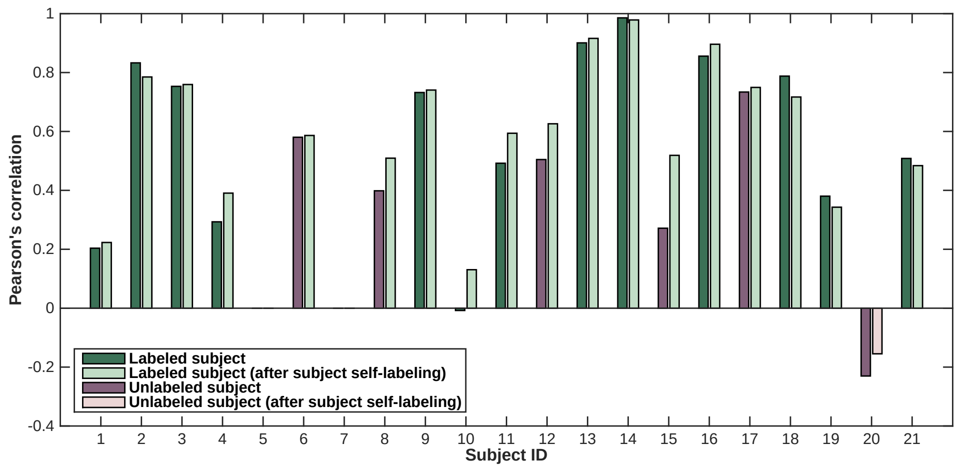


Figure 32. Pearson's correlation and labeling before and after subject self-labeling (FE2).

Classification performance on set *S**

The goal of subject self-labeling was to increase the labeled data from subjects in ***U*** to improve the performance of the frustration classification system presented in chapter 5. With the new set ***S****, and using LOSOCV, new ROC curves were produced (see Fig. 33 and Fig. 34). These new curves are very similar to the ones shown in chapter 5, and only an improvement of 1.95% and 2.44% has been obtained for FE1 and FE2, respectively. That is, new AUROC values are 0.8365 for FE1 and 0.7923 for FE2. Such small improvement is not very significant. However, it is important to note that the quantity of new data added (data from 2 subjects in FE1 and 5 subjects in FE2) is not very significant either. Better ROC curves are expected with the introduction of more data.

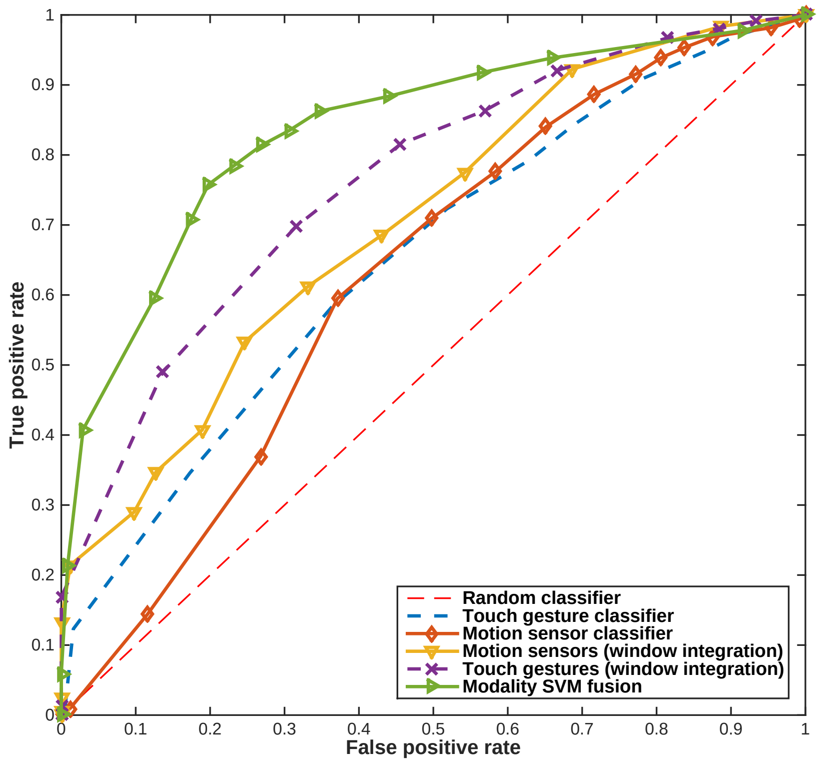


Figure 33. ROC curves during FE1 for single modality and modality SVM fusion classifiers, averaged after LOSOCV on semi-supervised set *S.**

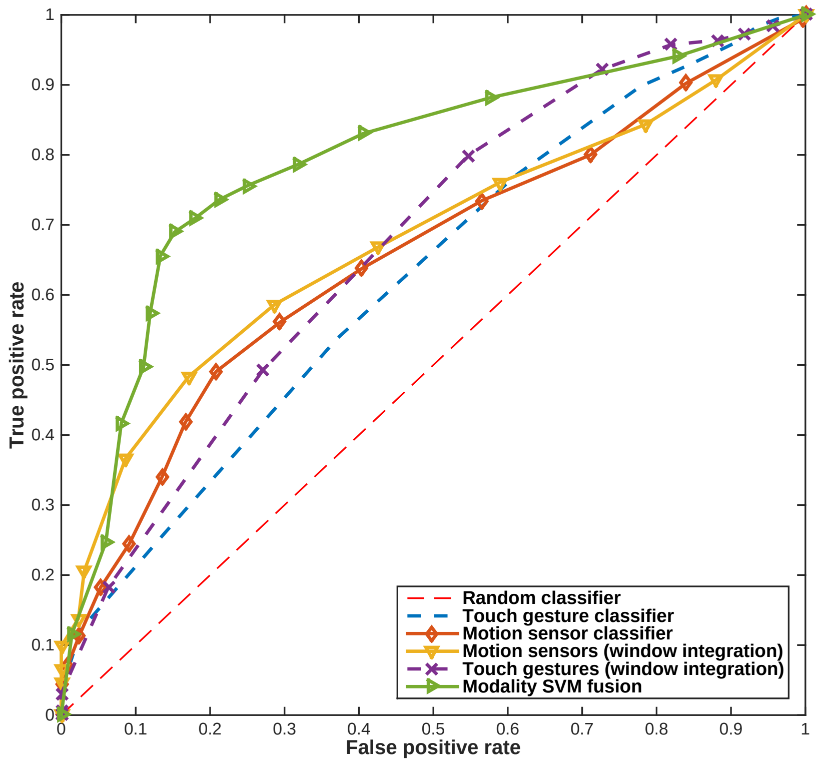


Figure 34. ROC curves during FE2 for single modality and modality SVM fusion classifiers, averaged after LOSOCV on semi-supervised set *S.**

Summary and conclusions

Based on the analysis of subject's video recordings, the dataset was divided into a supervised set ***S*** (corresponding to those subjects from which a frustrating reaction was observed), and an unsupervised set ***U*** with unknown labels. The problem was turned into a semi-supervised learning problem where set ***S*** is used to determine the labels of ***U***.

Clustering techniques such as *k*-means did not succeed in providing a good separation between the two classes in either a raw feature space or a dimensionality-reduced space using techniques such as PCA and deep autoencoders.

Instead, subject self-labeling was implemented using the frustration classification system described in chapter 5. During subject self-labeling, subjects were iteratively added to the supervised set if their corresponding correlation coefficient was higher than a threshold set to 0.5. However, some subjects pertaining to ***S*** still present a correlation value below 0.5 and so this indicates that the classification system presented does not fit every subject's data well. Furthermore, subject self-labeling might be reinforcing mistakes to the classifier.

Semi-supervised learning is usually applied when the number of labeled data is much lower than the number of unlabeled data. In this work, both the number of labeled and unlabeled data is small and it is therefore hard to guarantee good generalization to new data.

New ROC curves have been obtained using the new set ***S****, with a small improvement (around 2% improvement).

CHAPTER 7

DISCUSSION AND FUTURE WORK

With this work I aimed to contribute in the literature of frustration detection using a smartphone. To the best of my knowledge, this is the first attempt to detect frustration from a smartphone user using smartphone sensors alone. The intuition behind this work is that the way a smartphone user holds its device or touches the screen can potentially show signs of frustration, independently of the task.

This thesis aims to present a proof of concept in the detection of frustration, and the complexity of the models or other implementation details were not a major concern. However, the computing capacity of current smartphones suggests that implementing the proposed system could be possible. Note that most of the computing time is spent training the classifier, and this is done off-line with –ideally– a large dataset. Once a classifier is trained, the prediction step is usually much faster. On the other hand, feature extraction in real-time is needed. Being able to identify the most predictive features is an interesting extension of this work to ease real-time implementations. Note that motion sensor features in time were the most predictive features, and these are the simplest and least expensive ones to compute.

The goal is to detect frustration not just for the purpose of fixing a program error (which would be much easier to do from an OS perspective), but rather, to enhance the user experience and the design of applications by identifying frustrating actions or situations that are not errors by definition (e.g. copying a text from a web browser can become frustrating if the user cannot select the text that he/she wants – the text selection feature is then identified to be not very user-friendly).

Other potential applications include intelligent tutoring systems. It has been shown that learning rates are inversely correlated with student's frustration. An intelligent tutoring system that is able to detect frustration could provide an alternative way of teaching the material to try to reduce the student frustration levels. Researchers working on the development of smartphone applications for the blind community could also benefit from the feedback of identifying frustrating events in their applications.

I designed the experiment described in chapter 2 introducing program errors as an easy and efficient way to frustrate a user at a specific time. I was inspired by daily “errors” that usually frustrate people such as loading dialogs when watching and enjoying a video or losing information that was not saved. I believe this design is based on realistic scenarios (e.g. completing a form), which allows the user to naturally use the device when typing and holding the phone in both landscape and portrait orientations. The only constraint was that subjects were not allowed to place the smartphone on a surface such as a table to guarantee that valid sensor data was being acquired at all times in a non-intrusive way.

One important remark that I wanted to show is that self-frustration is not reliable and should not be used in similar work with respect to the study of frustration (as it has been widely done in the literature). While many subjects reported that they were not frustrated, their facial and body reactions showed the opposite. Even though self-reports were collected, it is important to clarify that they were only used to make such statement.

I believe that the ground truth of the experiment was set in a way that only the change of affective state was different between the control and frustrating conditions. Some important remarks on the experiment setup are:

- At the first stage of classification, data points are classified independently. Each data point corresponds to a single touch gesture or a 2-second window of sensor data. For this reason, long-term patterns from the multitasking game cannot be learned by the classifier.
- There are no substantial differences between tap sequences before and after FE1. For instance, there are many instances of three consecutive taps when the user hears “shake, shake, shake” from the song “Shake your booty”, and this sequence occurs both before and after the error. The experiment was carefully designed so that the same activity was performed before and after each frustrating event for a controlled ground truth.
- The amount of data collected during the survey is around 1 to 3 minutes before the onset of FE2 and 1 to 3 minutes afterwards, which leads to a 50/50 partition of the data among the two class labels. In future

experiments, it would be important to include more typing exercises prior to the date format error to increase the amount of data under control conditions. However, in the case of FE1, all data points from stage 2 (2nd multitask video with no errors) were added as control data, as well as the first half of stage 3 (prior to the onset of FE1). In this case, the amount of control data is higher.

- Data points immediately following each frustrating event were rejected. In the case of FE1, data points that occurred during the error and loading dialogs were rejected. Right after FE2, all subjects scrolled down and up again to confirm that all their answers had been deleted. Data points coming from this prior scrolling were discarded. By doing this, I tried to guarantee that there were no differences in the tasks before and after the error, except for the possible change of the affective state of the user.

Through this thesis, it has been assumed that the only affective state present was frustration, but this is a hard assumption as there can also be traces of amusement, confusion or interruption. Determining a user affective state is a delicate problem in human-computer interaction as well as in cognitive science, and I hope this works serves to incentivize more work towards this direction.

The results presented show that motion sensors and touch gestures extracted from a smartphone can potentially determine user frustration. Reasonable ROC curves have been obtained when the input features are integrated over a window of time and the two available modalities are fused using a SVM fusion rule. Adding more modalities such as keystrokes when typing or speech is expected to further increase performance in future work.

Further research is also needed to design a refined model of frustration for automated real-time prediction of frustration. Given the short period of each experiment stage, it has been assumed that frustration remained constant over time after the onset of each error. But frustration, as any other affective state, evolves and decays over time and so further research is needed to properly model this decay.

I acknowledge that a larger and more diverse group of subjects is needed to build a classifier that can attest generalization and that the size of this pilot study (21 subjects) is rather small. In the future, more efforts should be channeled towards extracting more data and analyzing the performance on a wider population.

REFERENCES

1. Siegrist, J., Menrath, I., Stöcker, T., Klein, M. et al. Differential brain activation according to chronic social reward frustration. *NeuroReport*, 16, 1899-1903, 2015.
2. Haner, C. F., & Brown, P. A. Clarification of the instigation to action concept in the frustration aggression hypothesis. *Journal of Abnormal Psychology*, 51(2), 204-206, 1955.
3. Kapoor, A., Burleson, W., Picard, R.W. Automatic prediction of frustration. *Int. J. Human-Comput. Stud*, 724-736, 2007.
4. Craig, S.D., Graesser, A.C., Sullins, J., Gholson, B. Affect and learning: An exploratory look into the role of affect in learning. *J. of Educational Media*, 2004.
5. S. L. Happy, A. Dasgupta, P. Patnaik, A. Routray. Automated alertness and emotion detection for empathic feedback during e-Learning. *The 5th IEEE Int. Conf. on Technology for Education*, 47-50, 2013.
6. Grafsgaard, J.F., Wiggins, J.B., Boyer, K.E., Wiebe, E.N., Lester, J.C. Automatically recognizing facial expression: Predicting engagement and frustration. *Proc. 6th Int. Conf. on Educational Data Mining*, 2013.
7. Z. Zeng, M. Pantic, G. I. Roisman, T. S. Huang. A survey of affect recognition methods: audio, visual, and spontaneous expressions. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 39-58, 2009.
8. A. Rabie, B. Wrede, T. Vogt, M. Hanheide. Evaluation and discussion of multi-modal emotion recognition. *2nd Int. Conf. Computer and Electrical Eng.*, 598–602, 2009.
9. Hoque, M.E., Picard, R.W. Acted vs. natural frustration and delight: Many people smile in natural frustration. *9th IEEE Int. Conf. on Automatic Face and Gesture Recognition*, 2011.
10. S. D’Mello, A. Graesser. Automatic detection of learner’s affect from gross body language. *Applied Artifical Intelligence*, 123-150, 2009.
11. R. Fernandez, R. Picard. Signal processing for recognition of human frustration. *IEEE Int. Conf. on Acoustics, Speech and Signal Processing*, 1998.
12. Liao, W., Zhang, W., Zhu, Z., Ji, Q., & Gray, W. D. Toward a decision-theoretic framework for affect recognition and user assistance. *International Journal of Human-Computer Stud*, 847-873, 2006.
13. J. McCuaig, M. Pearlstein, A. Judd. Detecting learner frustration: towards mainstream use cases. *Intelligent Tutoring Systems*, 21-30, 2010.
14. Y. Qi, C. Reynolds, R. W. Picard. The Bayes Point Machine for computer-user frustration detection via Pressure-Mouse. *Proc. Perceptive User Interfaces*, 2001.

15. E. Miluzzo, A. Varshavsky, S. Balakrishnan, R. Choudhury. TapPrints: your finger taps have fingerprints. *Proc. Mobile Systems, applications and service*s, 2012.

16. Kwapisz, J.R., Weiss, G.M., and Moore, S.A. Cell phone-based biometric identification. *Biometrics*, 2010.

17. Wolff, Matt. Behavioral biometric identification on mobile devices. *Springer Berlin Heidelberg*, 2013.

18. K. Choi, K. A. Toh, H. Byun. Real-time training on mobile devices for face recognition applications. *Pattern Recognition* 44(2), 386-400, 2011.

19. E. Vasiete, Y. Chen, I. Char, T. Yeh, V. M. Patel, L. Davis, R. Chellappa. Toward a non-intrusive, physio-behavioral biometric for smartphones. *MobileHCI*, 501-506, 2014.

20. I. Laptev, M. Marszalek, C. Schmid, B. Rozenfeld. Learning realistic human actions from movies. *Proc. CVPR*, 2008.

21. M. Frank, R. Biedert, E. Ma, I. Martinovic, D. Song. Touchalytics: on the applicability of touchscreen input as a behavioral biometric for continuous authentication. *IEEE Transaction On Information Forensics and Security*, 2013.

22. Android developers website: developer.android.com/

23. Burrus, C. S., R. A. Gopinath, H. Guo. Introduction to wavelets and wavelet transforms, a primer. *Prentice Hall*, 1998.

24. Khushaba, A. Al- Jumaily, and A. Al-Ani. Novel feature extraction method based on fuzzy entropy and wavelet packet transform for myoelectric control. *7th Int. Symposium on Communication and Information Technologies*, 352-357, 2007.

25. Freedman, David. Statistical models: Theory and practice. *Cambridge University Press*, 2009. ISBN 978-0-521-67105-7.

26. Bishop, Christopher M. Pattern recognition and machine learning. *Springer*, 2006. ISBN 978-0387-31073-2.

27. Bengio, Yoshua. Learning deep architectures for AI. *Foundations and trends in Machine Learning*, Vol: 2, 1-127, 2009.

28. Breiman, Leo. Random Forests. *Machine Learning* 45(1), 5-32, 2001.

29. Boser, B. E., Guyon, I. M., Vapnik, V. N. A training algorithm for optimal margin classifiers. *Proc. of the fifth annual workshop on Computational learning theory*, p. 144, 1992.

30. Srivastava, N., Hinton, G., Krizhevsky, A, Sutskever, I. and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *JMLR*, p. 30, 2014.

31. Erhan, D., Bengio, Y., Courville, A., Manzagol, P. A., Vincent, P., and Bengio, S. Why does unsupervised pre-training help deep learning? *The Journal of Machine Learning Research*, 625-660, 2010.