TÉCNICO LISBOA

Physiologically Attentive User Interface for Robot Teleoperation

António Jorge Pires Gomes Tavares

Thesis to obtain the Master of Science Degree in

Aerospace Engineering

Supervisors: Prof. Rodrigo Martins de Matos Ventura
Prof. José Luís da Silva

Examination Committee

Chairperson: Prof. José Fernando Alves da Silva
Supervisor: Prof. Rodrigo Martins de Matos Ventura
Member of the Committee: Prof. Carlos António Roque Martinho

October 2020

Acknowledgments

First and foremost, I would like to thank Professor Rodrigo Ventura and Professor José Luís Silva for giving me the opportunity to integrate this project. For their insight, continuous support and sharing of knowledge that made this thesis possible.

I would also like thank José Corujeira and Rute Luz for their constant support and for always being available to help me whenever I needed.

Besides, a special thank you to all the people that participated in the experimental sessions and were willing to help even in times of pandemic.

I would also like to thank my friends Cat, Ema, Henri, Inês, Menezes, Néné, Rocha, Saraiva, Sofia and Tavares for making my university years such a great journey, always supporting me in the good and bad moments in my life. I hope that we will continue to be a part of each other's lives for many more years. A heartfelt thank you to Cat, Henri, Jacob, Moita and Rita for being the best roommates I could ever ask for. A sincere thank you to Alex, Baptista, Cordeiro, Edu and Ema for being at my side for all these years, both in and out of Lisbon. I would also like to extend my gratitude to Tuna Académica de Lisboa and all the people that are part of it, for showing me that music is even more beautiful when made with friends.

I also want to express my deepest gratitude for Cat. Thank you for the unconditional support you have always given me, that made such a huge difference. This work would have been impossible without you by my side. Thank you for the love and friendship that we shared and will continue to share for many more years.

Last but not least, I would like to thank my parents and sister for their encouragement and caring over all these years. Without them, I would not be the person I am today and would have never reached this point in my life.

To each and everyone of you, my warmest thank you.

Abstract

The management of user attention is becoming a crucial challenge in the development of modern user interfaces, both in the Human-Computer Interaction and Human-Robot Interaction fields. User interfaces are shifting from being information-hungry devices to being attentive systems that consider their user's needs upon interaction. The interfaces developed for robot teleoperation can be particularly complex, often displaying very high amounts of information on their screens, which can induce a great deal of cognitive overload on the operators during life-critical missions. In this dissertation, a prototype for a Physiologically Attentive User Interface is presented, which is applied to an Urban Search and Rescue robot that provides a complex user interface. The system analyses physiological data, facial expressions, and eye movements to classify three emotional states (rest, stress, and workload) during robot teleoperation tasks. An attentive user interface is then assembled, which is modified dynamically according to the predicted emotional state in order to manage the user's focus during mentally demanding situations. This work contributes with the design of a user experiment, comprising emotion induction tasks that successfully trigger high and low cognitive overload states, along with effective strategies of managing user attention. Results from a user evaluation revealed no statistically significant differences in terms of the task performance and usability achieved when comparing this system to a classic user interface. However, the results were limited by the small number of subjects available for the study and the poor performance of the emotional state classifier.

Keywords

Robot Teleoperation; Attentive User Interface; Emotion Classification; Neural Networks.

Resumo

A gestão da atenção do utilizador tem vindo a tornar-se num fator essencial a ter em conta no desenvolvimento de interfaces de utilizador modernas. Estes sistemas estão cada vez mais a tornar-se em sistemas atenciosos que têm em consideração as necessidades dos seus utilizadores. As interfaces desenvolvidas para a teleoperação de robôs podem ser particularmente complexas, mostrando grandes quantidades de informação de forma pouco clara, o que pode induzir uma grande sobrecarga cognitiva nos seus operadores. Nesta dissertação, é apresentado um protótipo de uma Interface de Utilizador Fisiologicamente Atenta, que é aplicada a um robô de busca e salvamento que fornece uma interface de utilizador complexa. O sistema analisa dados fisiológicos juntamente com expressões faciais e rastreamento ocular para classificar três estados emocionais (repouso, stress e carga de trabalho) durante o decorrer da teleoperação do robô. O estado emocional previsto é utilizado para modificar a interface dinamicamente, de modo a gerir o foco do utilizador durante situações mentalmente exigentes. Este trabalho contribui com a concepção de um estudo com utilizadores, compreendendo tarefas que desencadeiam com sucesso estados de baixa e alta sobrecarga cognitiva, juntamente com estratégias eficazes de gestão da atenção do utilizador. A avaliação deste sistema não revelou diferenças estatisticamente significativas em termos do desempenho e usabilidade alcançados com este sistema, em comparação com uma interface de utilizador clássica. No entanto, os resultados foram limitados pelo pequeno número de sujeitos disponíveis para o estudo e pelo baixo desempenho do classificador do estado emocional.

Palavras Chave

Teleoperação de robôs; Interface de Utilizador Atenta; Classificação de Emoções; Redes Neuronais.

Contents

List of Figures

List of Tables

Acronyms

AUI	Attentive User Interface
BMI	Brain-Machine Interface
CFS	Correlation-based Feature Selector
DBN	Deep Belief Network
DEQ	Discrete Emotions Questionnaire
DL	Deep Learning
ECG	Electrocardiography
EDA	Electrodermal Activity
EEG	Electroencephalography
EMG	Electromyogram
GUI	Graphical User Interface
HCI	Human-Computer Interaction
HMM	Hidden Markov Model
HRI	Human-Robot Interaction
HRV	Heart Rate Variability
IQR	Inter-Quartile Range
ITQ	Immersive Tendencies Questionnaire
KNN	K-Nearest Neighbour
LDA	Linear Discriminant Analysis
LDS	Linear Dynamic System
ML	Machine Learning
MLP	Multi-layer Perceptron

MTU	Maximum Transmission Unit
PAUI	Physiologically Attentive User Interface
PCA	Principal Component Analysis
SVM	Support Vector Machine
TCP	Transmission Control Protocol
t-SNE	t-distributed Stochastic Neighbour Embedding
UDP	User Datagram Protocol
URL	Uniform Resource Locator
USAR	Urban Search and Rescue

1

Introduction

Contents

In this chapter, the context of the problem addressed in this thesis and the reasons that drove its study are presented. Afterwards, the research statement of the thesis is declared, along with the hypothesis formulated for its evaluation. Additionally, the contributions made by this work to the scientific community are described. To close the chapter, the underlying structure of the document is provided.

1.1 Contextualization and motivation

Throughout the last few decades, society has witnessed a sharp rise in the level of technology available to humans. Not only the number of displays per human has increased substantially, but also the computational capabilities of our devices, such as PCs, laptops, smartphones, tablets, amongst many others, are now greater than ever. In 2010, the number of network connected devices per person around the world was estimated to be 1.83, while it is expected to be 9.27 in 2025 [1]. As the general population is becoming more and more surrounded by numerous interfaces that constantly distract us with pop-ups, such as notifications or messages, the attention span of users is getting reduced to the point where they have to filter information from multiple sources, often with the drawback of doing it at a superficial level, effectively limiting their ability to interact properly with each system [2]. This problem brings in the need for smarter interfaces that understand human needs and adapt to them, namely in the Human-Computer Interaction (HCI) field.

In order to tackle this problem, Vertegaal (2003) suggested a model for an Attentive User Interface (AUI), designed to be sensitive to the user's attention and act accordingly. These attentive user interfaces take advantage of overt properties of user attention, such as user presence, proximity and gaze direction, to determine which task or device the user is focused on and, consequently, his availability for interruptions [3]. However, while these measurements can help the understanding of the current visual focus of the user, they do not provide information about the state of mind, which can be just as or more relevant, since the user's physical activity is not necessarily an indicator of mental engagement. Fortunately, due to scientific advances in the field of Psychophysiology, it is possible to establish links between the human body and mind in order to acquire more reliable information about human cognitive and emotional states. As human physiology is considerably influenced by the activity of the Central Nervous System (CNS) and the Autonomic Nervous System (ANS), a change in the state of mind reflects itself as a series of physical signals in the body [4]. For instance, a user under stress will likely experience increased heart rate and blood pressure. Therefore, the measurement of physiological signals allows the perception of covert states of mind that are not visible to an AUI. With this in mind, Chen (2006) proposed a framework for a Physiologically Attentive User Interface (PAUI) that resorts to physiological measures to respond actively to the user's needs. The use of a PAUI effectively extends the reach of an AUI, in the sense that it enables a deeper understanding of the emotional state of the user [5].

With the evolution of robotics and the increased extent of interactions between humans and robots, the problem of managing user attention also expanded to the field of Human-Robot Interaction (HRI), particularly in the robot teleoperation area. The ever-growing sophistication of the tasks performed by robots often comes with the drawback of requiring operators to deal with very complex interfaces that can potentially submit them to a great deal of stress and workload, compromising their focus and performance on the given task. This issue is especially problematic in fields like Urban Search and Rescue (USAR) operations, where it is essential to guarantee that the operator's focus remains at its best, which would otherwise leave a person's life at stake.

In recent years, with the growth of computational power, the analysis of data and its employment in the development of artificially intelligent systems has led to increasingly effective algorithms for data classification that yield a vast array of applications, from self-driving cars, to medical diagnosis assistance, and even emotion recognition. The growing burst of available data is giving these algorithms more potential than ever, particularly in the field of Machine Learning (ML) and, more specifically, Deep Learning (DL). This promotes the development of intelligent systems that can accurately learn useful properties of data that are often not detectable by humans.

This dissertation presents the development of a PAUI that aims to tackle the problem of managing user's attention and focus during robot teleoperation tasks. In this case, the PAUI was specifically applied to a USAR robot called RAPOSA. The PAUI employs the classification of the user's emotional state in real-time based on neural networks, whose predictions are used to assemble an AUI on top of the original interface without accessing its source code. This thesis was elaborated upon the framework defined by Singh et al. (2018) [6] for a PAUI applied to this specific robot.

1.2 Research Statement

The purpose of this thesis is to develop a PAUI applied to the robot teleoperation field. The PAUI allows the retrieval of physiological data from the user to understand covert states of mind, along with overt measurements of user attention, such as eye tracking and facial expressions, in order to detect the user's emotional state between three different states (rest, stress and workload) with the aid of DL techniques. The emotional state prediction is then used to dynamically change the interface in real-time in order to improve the user's focus during high cognitive stress moments. The PAUI is established over the preexisting robot Graphical User Interface (GUI) with the purpose of reducing its complexity and displaying information in a clearer way, thus decreasing the emotional overload of the operator. Furthermore, the PAUI interacts with the old GUI by means of an automation tool and extracts its image in order to render an AUI based on the original interface, thus eliminating the need to develop a new interface or a completely new solution from scratch when the source code is not available.

The objective of this work is to understand if the employment of a PAUI in this specific context can enhance the usability of the interface when operators are under high cognitive stress and workload conditions, while improving their ability to stay focused when compared to a classic GUI approach. The ISO 9241-11 (Ergonomics of human–system interaction) defines usability as the ”extent to which a system, product or service can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use”. Here, effectiveness refers to the ”accuracy and completeness with which users achieve specified goals”, efficiency refers to the ”resources used in relation to the results achieved” and satisfaction refers to the ”extent to which the user's physical, cognitive and emotional responses that result from the use of a system, product or service meet the user's needs and expectations” [7].

Thus, the research statement of this dissertation can be highlighted as follows:

> *When submitted to high cognitive stress or workload situations in robot teleoperation tasks, the usability of the system and the operator's ability to stay focused can be increased by dynamically changing the interface with respect to the measured emotional state.*

Taking this into account, the research aims to assess the following hypotheses:

H1: *The employment of a PAUI in robot teleoperation tasks improves the efficiency of operators in comparison with a classic GUI approach.*

H2: *The employment of a PAUI in robot teleoperation tasks improves the effectiveness of operators in comparison with a classic GUI approach.*

H3: *The employment of a PAUI in robot teleoperation tasks improves the operator's ability to remain focused during missions in comparison with a classic GUI approach.*

H4: *The employment of a PAUI in robot teleoperation tasks improves the level of usefulness and satisfaction experienced in comparison with a classic GUI approach.*

1.3 Contributions

Taking into consideration the problem of managing user's attention in the context of robot teleoperation, this work offers the following contributions:

- **Prototype of the PAUI**

  This work introduces a prototype of a PAUI for robot teleoperation, offering details about the strategies implemented to manage user's attention, the techniques adopted in the development of the emotional state classifier, as well as the methodology undertaken to assemble these components

together to create a PAUI on top of the pre-existing robot's interface. Although being applied to the RAPOSA's interface, the system can be adapted to other interfaces (see section 3.4).

- **Design of emotion induction tasks**

  The necessity of stimulating different emotions on the users resulted in the planning of tasks that successfully induced three distinct emotional states (rest, stress and workload). These tasks fall within the USAR context, accurately representing the circumstances commonly faced by operators in this field.

- **USAR Simulator**

  The conditions of the experiment (see section 6.1) led to the development of a Search and Rescue simulator that enabled the re-creation of the robot's main functionalities and features, along with the modelling of home-like environments that depict real Search and Rescue missions.

- **User study**

  The reports of the user study carried out in order to evaluate the research statement hypotheses are described, including a detailed procedure of the experimental sessions, followed by the results obtained and the respective discussion.

1.4 Structure

The contents of this dissertation are organized as follows:

First chapter: The thesis is contextualized and the motivation behind it is presented. The research statement is declared, along with the hypotheses formulated and the contributions offered by this work. Additionally, the dissertation structure is outlined.

Second chapter: Related work that motivated the thesis is described, as well as existent research in the fields of study that aided its development. Alongside, the state-of-the-art solutions to the problem are presented together with reasoning as how they have driven the work presented in this dissertation.

Third chapter: The approach adopted in order to assemble the PAUI is outlined, explaining the modules that can be identified from the overall architecture of the system.

Fourth chapter: An overview of the background concepts behind the development of the neural network classifier is presented, along with key choices and methodologies that were adopted in its implementation.

Fifth chapter: The attentive aspect of the interface is described, explaining how it was built on top of the pre-existing interface through means of image extraction and task automation.

Sixth chapter: The evaluation procedures carried out for the validation of the proposed approach are presented, along with the results obtained.

Seventh chapter: The conclusions drawn from this research are presented, highlighting the contributions it offers and answering the main research question, along with the presentation of recommendations for future work.

2

Related Work

Contents

In this chapter, an overview of the present state-of-the-art technologies that were relevant in the development of this work are presented. First, a review of the important characteristics attributed by researchers to the design of attentive user interfaces are described, along with some examples of applications that take into consideration the user's needs and intentions. Afterwards, an examination of studies that delve into the recognition of human emotional states is given, showing the various types of data that are commonly used to classify emotions, along with the results obtained for these methods. Emphasis is also placed on procedures adopted by researchers to improve the classification accuracies of emotion recognition models, since the achievement of good accuracies on emotion classification can still prove to be a difficult task. Finally, a brief discussion of the reviewed research and its influence on the development of this work is given, along with the reasons that motivated its conception.

2.1 Attentive User Interfaces

The concept of an AUI has been target of significant concern as the years go by. The massive growth in the number of computing devices and interfaces available to humans leads to a need for interfaces that understand and adapt to human emotions and needs. Vertegaal (2003) [3] proposed a framework for the development of an AUI, i.e. an interface that is sensitive to the user's necessities, which can be achieved through the measurement of covert characteristics of user attention, such as user presence, gaze direction, proximity and speech. These interfaces can then act on this information and decide when the user is available for interruptions, delivering them in a progressive way instead of forcing the information upon the user, potentially leading to a decrease in the user's focus. AUI's can also enhance important elements of an interface that the user tends to focus on, while alleviating the less important features of the interface (see figure 2.1). Taking these factors into consideration, the framework proposed by Vertegaal et al. (2006) [8] for the development of AUI's comprises five key characteristics:

- **Attention sensing:** Attentive interfaces can detect the task, device or person the user is focused on through the measurement of covert properties of user attention;
- **Attention reasoning:** After determining where the user's focus is directed to, attentive interfaces can establish an optimal order of priorities for the tasks to be fulfilled by the user;
- **Attention communication:** The information inferred about the user's attention levels is communicated to other devices and people;
- **Negotiation of turns:** Devices should check the task priorities defined and gradually signal the user of a new interruption;
- **Focus augmentation:** The employment of an AUI leads to the improvement of the user's concentration by emphasizing the most relevant information.

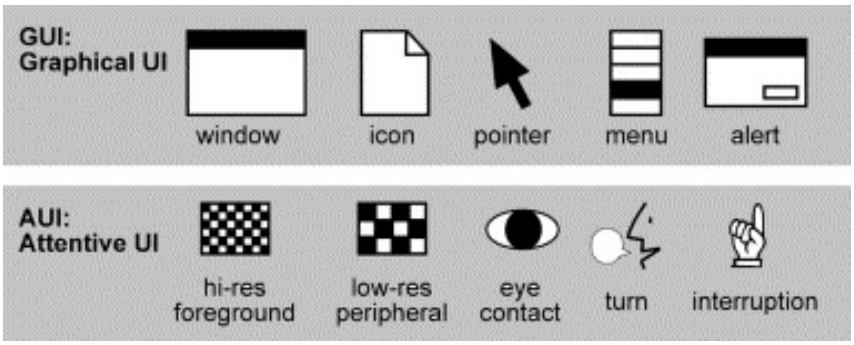


Figure 2.1: Interaction style of an AUI compared to a classic GUI. [8]

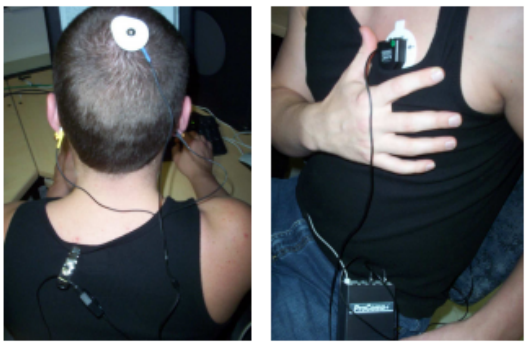
Bulling (2016) [2] considered the management of user attention as a *"critical challenge for next-generation human-computer interfaces"*. Researchers in the HCI field are becoming aware of the fact that human attention and focus is a limited resource that can play a very important role in the performance of the interfaces themselves [8]. Bulling addressed the problem of *continuous partial attention*, stating that the shifting of focus between various sources of information can effectively lead to a reduction in the overall focus of the user, since it limits the ability to concentrate on a specific task. Due to the advancements in the accuracy of technologies that can accurately measure user's attention, Bulling defined six categories that should be taken into account in the development of a new generation of *pervasive attentive user interfaces*:

- **Unobtrusiveness:** In order for AUI's to be introduced in the everyday life of its users, discreet methods to measure the user's attention should be considered;
- **Accuracy:** The assurance of accurate estimations of user attention is vital to the implementation of such systems;
- **Large scale:** Apart from assessing the attention level of individuals, future interfaces should also measure the attention dynamics of large groups;
- **Long-livedness:** Future pervasive interfaces should model the user's attention over large periods of time, over the course of everyday life activities;
- **Seamlessness:** The acquisition of information on the user's attention levels should be carried out by multiple sensors, transitioning seamlessly between them;
- **Context awareness:** Modern attentive interfaces should employ contextualized methods of estimating user attention that meet the interface's objectives.

Bulling concludes that the development of new techniques that allow the accurate assessment of a user's attention and needs can effectively lead to an improvement of the user's performance, as well as the interface's usability [2].

One example application of an attentive interface is Project Aura, a system developed by Garlan et al. (2002) [9] that intends to reduce the burden enforced on the user in environments that involve the usage of wireless communication in wearable or handheld devices. Aura can work in conjunction with smart spaces in order to figure out the bandwidth in certain areas and automate tasks that are prioritized by the user when he/she is perform other activities, such as walking or driving. Other research also suggests the use of eye and body movements along with facial expressions as input for user interfaces, enabling a more fluid and effortless interaction [10].

While AUI's can accurately detect if a user is paying attention to a certain device through the recognition of the user's gaze direction, they cannot determine the actual level of engagement of the user with that device, i.e. if the user is actually focused or not on performing some specific task. This limitation arises from the fact that AUI's rely on overt means of measuring a user's attention. For this reason, Chen and Vertegaal (2004) [11] proposed a prototype for a PAUI that bases its model of interruptibility on covert measurements of the user's mental load, such as physiological signals. The PAUI presented by Chen and Vertegaal makes use of LF (Low Frequency) spectral components derived from Heart Rate Variability (HRV) signals to measure mental load. Additionally, the analysis of an electroencephalogram enables an understanding of the user's motor activity (see figure 2.2(a)). These signals allowed the classification of the user's mental and motor activity in order to differentiate four distinct user states that can be used to predict the user's availability for interruptions. In this case, the PAUI regulates the notifications issued by a cellphone (see figure 2.2(b)), depending on the predicted state and the cost of interruptions for each state, which is defined by the user's preferences relative to the phone's notification level (ring, vibrate or silent) and to the availability for e-mail notifications. The first state usually reveals the lowest level of engagement, where the user is at rest and shows signs of low mental activity. For this reason, the cost of interruptions is also very low, since the user is in a relaxed state. In the second state, the user exhibits high motor activity and low mental load, which is common when the user is moving between places. Here, the cost of interruptions is low for speech-related tasks such as answering phone calls, but is high for movement-related activities, such as writing e-mails. The third state is the opposite of the second state, where the user is undergoing mental effort but is not moving, such as during a reading or driving session. In this case, the user might still want to be notified, but not through audible means that might interfere with the task that is being performed. The fourth state has the highest cost of interruptions, since the user is going through both mental and physical efforts. During these tasks, users might not be available at all for any kind of interruption, since it could potentially consume mental resources that are necessary for the task at hands. The application of a PAUI to the management of the notifications issued by a cellphone proved to be successful in handling the user's focus with respect to different emotional and physical states, which were predicted with an accuracy of 83%.



(a) Sensors used for measuring user's mental and physical load (electroencephalogram on the left and electrocardiogram on the right).

(b) User testing the PAUI applied to the management of notifications on a cellphone.

Figure 2.2: Apparatus used and testing in the PAUI developed by Chen and Vertegaal (2004). [11]

Significant research has also been done in improving user experience in the HRI field. Guo and Sharlin (2008) [12] developed a technique for capturing human arm movements and hand gestures that were used as input to control a robot in navigation and posture tasks, revealing an improvement in task performance when compared to the original keypad controls. Millan et al. (2004) [13] employed a Brain-Machine Interface (BMI) based on non-invasive electroencephalogram analysis in conjunction with advanced robotics to achieve brain-actuated control of a robot. In this research, a robot was successfully driven from one room to another relying purely on mental commands, proving that the task of mentally operating robots in indoor environemnts is almost as efficient as the classic manual control. This opens up possibilities for applications that aim to help physically disabled people, such as mentally controlling a wheelchair. Similarly, the Honda Research Institute Japan, Advanced Telecommunications Research Institute International (ATR), in cooperation with the Shimadzu Corporation [14], developed a BMI for the operation of a robot by human thoughts only (see figure 2.3). By measuring electric potential differences on the scalp through Electroencephalography (EEG) and brain blood flow changes with near-infrared spectroscopy, the human thought process was captured and processed, allowing a user to imagine body movements that were then similarly replicated by a robot with an accuracy of 90%.

Figure 2.3: ASIMO humanoid robot replicating human body movements that are derived from the analysis of the user's thought process. [14]

The necessity for improvement of user interfaces and the simplification of the respective user interaction style has also been manifested in USAR operations. Human teams that operate in these conditions are required to scout damaged structures for survivors, often with the assistance of teleoperated robots that can easily access areas that might be too dangerous for a human to investigate. However, the user interfaces provided for the teleoperation of these robots usually contain large amounts of information about the robot's state and the environment. Baker et al. (2004) [15] studied more than a dozen USAR robot interfaces used in the American Association for Artificial Intelligence (AAAI) and the RoboCup Robot Rescue competition, identifying the strengths and weaknesses that are common amongst them. This study found out that, in USAR missions, operators tend to focus largely on the area of the interface that displays the video feed, disregarding the rest of the information presented on the interface for most of the time, since a large part of it might not be relevant for a specific task or person. These elements of the interface can potentially take valuable space that could be used for augmenting the display of useful information. Baker et al. (2004) used this knowledge to define enhanced awareness, lower cognitive load, increased efficiency and robot modality assistance as the main guidelines to take into consideration when designing a user interface for HRI in USAR environments. By employing these ideas in the re-design of a pre-existing interface, it was concluded that users found the new interface easy to use, although some experienced operators might miss some of the information that was removed [15]. Riley and Endsley (2004) [16] also expressed a concern for the lack of situational awareness in USAR operations, and emphasized the importance of the relationship between robots and operators used in these missions. Riley and Endsley identified the workload induced due to a visually demanding task and poor integration of data on interfaces as some of the most problematic causes of degradation of task performance in search and rescue operations. This study is concluded by pointing that user interfaces developed for USAR robots are often too complex for non-expert operators to use them effectively, emphasizing the need for new interface designs to take user experience into consideration and pursuit improved situational awareness [16].

2.2 Emotional State Estimation

The recognition of a user's emotional state can be a very useful input in the design of modern HCI and HRI systems, as it enables the development of affective computing strategies. A commonly used approach in the recognition of human emotions is through facial expression analysis. Anderson and McOwan (2006) [17] presented a facial expression recognition system that can distinguish the emotions of happiness, sadness, surprise, disgust, fear and anger, since those emotions are associated with unique facial expressions. This approach combines a face tracking algorithm with a Support Vector Machine (SVM) to classify the emotion portrayed by the user's face in real-time, achieving a successful

recognition rate of 81.82%. Hupont et al. (2013) [18] conceived an emotion recognition system based on facial expressions that operates in a continuous 2D emotional space, allowing for a vast array of intermediary emotions to be recognized. This system extracts facial features based on distances and angles between multiple face points (see figure 2.4), employing a combination of five commonly used classifiers (RIPPER, Multi-layer Perceptron (MLP), SVM, Naive Bayes and C4.5) to predict an emotion out of seven possibilities (disgust, joy, anger, fear, sadness, surprise and neutral). An accuracy rate of 73.73% was achieved in restrictive conditions, rising to 94.12% when evaluated under more relaxed criteria, in terms of whether the output of the system can be considered a success or not.

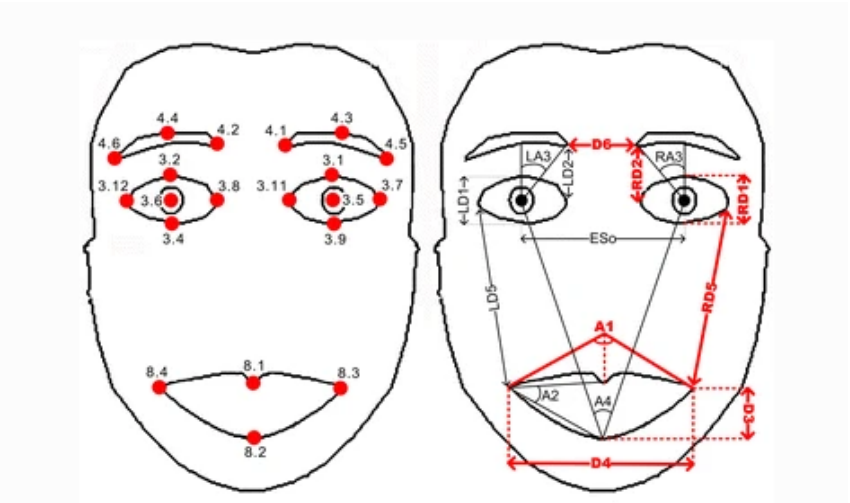


Figure 2.4: Tracked facial points and respective distances and angles that are extracted as features. [18]

With the growth of more accurate means of measuring physiological signals, the employment of such measurements is proving to be a another good source of information on the emotional state of a person. Agrafioti et al. (2011) [19] studied the feasibility and limits of using Electrocardiography (ECG) signals for differentiating emotional states. The process of extracting useful features from the ECG signal was preceded by the design of a synthetic signal whose main waves are synchronized with the captured ECG signal, followed by the decomposition of the signal in *Intrinsic Mode Functions* that represent different oscillation modes. The extracted features were then used to differentiate between passive and active arousal modes that are subject-dependent. Each subject was submitted to visual stimuli (passive arousal) that induced positive or negative sensations in one session, whereas in a second session they had to play an increasingly difficult video game (active arousal). Results showed a 26.02% lower arousal detection in the visual stimuli experiment, when compared to the video game experiment, showing the intrinsic capability of ECG signals to contain information that can lead to the differentiation of emotional states.

Other studies have also demonstrated the potential of peripheral physiological signals to exhibit

changes with respect to emotional states. Wagner et al. (2005) [20] acquired physiological data from subjects in four different emotional states in order to test the pattern recognition capabilities of three classifiers: linear discriminant function, K-Nearest Neighbour (KNN) and a MLP. Subjects were asked to pick a piece of music that represents, in their vision, each of the four affective states (joy, anger, sadness and pleasure), which was then used to induce that specific state during the recording of ECG, Electromyogram (EMG), skin conductivity and respiration rates. In order to improve classification accuracy, feature selection and reduction methods were applied, such as Principal Component Analysis (PCA) and Fisher projection, which enable a dimensionality reduction of the feature space, effectively increasing the classification accuracy from 80% to 92%. Another study carried by Kim et al. (2004) [21] performed emotion prediction on 50 subjects, all of which were children aged between 7 and 8 years old, in order to classify four emotional states (sadness, anger, stress and surprise) while acquiring data from ECG, Electrodermal Activity (EDA) and skin temperature variation. This study showed the clusters formed by each class had large variance within themselves, and significantly overlapped each other. A SVM was chosen as the classifier, obtaining a prediction accuracy of 61.76% for 4 classes and 78.43% when classifying only three classes (sadness, anger and stress).

Apart from the capability of the autonomous nervous system to convey information related to the emotional state of a person, which can be captured using biosignal sensors such as ECG, EMG and EDA, the central nervous system is also relevant in the study of emotion recognition based on physiological signals, since the brain is the origin of human emotions. Petrantonakis et al. (2011) [22] studied the effectiveness of emotion elicitation procedures based on EEG acquisition. The analysis of multidimensional directed information between the two brain hemispheres, allied with the frontal brain asymmetry theory led to the definition of an asymmetry index, which can effectively indicate which acquired signals have more valuable emotional information in relation to others. The conducted classification experiment employed an SVM that obtained a classification accuracy of 62.58% when using an user-independent setup and 94.40% in the user-dependent case. Similarly, Wang et al. (2014) [23] investigated the potential use of EEG features for emotion classification by conducting a series of experiments that induced positive or negative emotions in six subjects who watched movie clips that targeted specific emotions. The pre-processing of the acquired EEG signals led to to the extraction of three types of features, namely the power spectrum, wavelet and nonlinear dynamic analysis. Afterwards, the Linear Dynamic System (LDS) technique was applied to smooth the features and remove unwanted noise, followed by a dimensionality reduction using three different methods (PCA, Linear Discriminant Analysis (LDA) and Correlation-based Feature Selector (CFS)). The classification was performed using an SVM, obtaining the best average accuracy of 91.77% when using LDA for dimensionality reduction. This study also found that the power spectrum features (with an emphasis on higher frequency bands) provide the most robustness in classification out of the three types of features discussed. Zheng et al. (2014) [24] studied

the usage of advanced DL models to classify emotional data from two classes (positive and negative) relying on extracted 62-channel EEG signals from subjects exposed to emotion-inducing video clips. This study compared the performance of a Deep Belief Network (DBN) with an integrated Hidden Markov Model (HMM) and a single DBN to common approaches such as Graph-regularized Extreme Learning Machine (GELM), SVM and KNN, obtaining an accuracy of 87.62%, 86.91%, 85.67%, 84.08%, and 69.66% respectively, showing that DBN models outperform the other methods.

2.3 Discussion

Taking into consideration the state-of-the-art methodologies described previously, it can be assumed that a new generation of user interfaces that are attentive to their user's needs are emerging. User's attention is a precious resource that can take its toll on the interface's performance when put to the limit [2] [3]. The framework for these types of interfaces suggests the usage of eye movements and user presence as some important factors to consider when designing an AUI. The problem of managing user attention is manifesting itself in a wide range of areas, and has affected particularly the performance of USAR teams that use robots in their missions, due to the difficulty and complexity that comes associated with operating these robots [15] [16]. User interfaces developed for use in these robots often express a developer's point of view and do not consider the cognitive load forced upon its operators, or simply present too much information that is not relevant for most of the course of the mission, creating a high visual demand on the users. Operators may need to perform rescue operations during extended periods of time while interacting with these systems, leading to a degradation of their task performance, which can be critical in the context of scouting survivors. With the rise of more sophisticated artificially intelligent systems, emotion recognition is becoming possible to perform with relatively high precision due to the extraction of physiological signals [19] [21] [22] and facial expressions [17] [18], which can potentially convey valuable information on a user's state of mind. The assessment of a user's emotional state can be very useful in determining when a USAR robot operator is going through a high cognitive load, which is prone to happen in search and rescue missions. It is believed that this information can be used to improve the operator's experience by updating the robot's interface with respect to his/her emotional state, leading to a decrease of the workload induced and an increase of task performance.

3

Approach

Contents

This chapter introduces the PAUI approach, highlighting its core aspects and explaining the way each module of the approach interacts with each other. First, the solution found to the problem of managing the user's attention in the field of robot teleoperation is described, offering an overview of the full architecture of the approach, in order to contextualize the methodology that was carried out to develop the solution, which is further described in chapters 4 and 5. Afterwards, the data extraction and processing techniques used to acquire biometric signals that later enable the classification of an emotional state are explained. Subsequently, emotional state classifier adopted in this work is presented, followed by the interface manipulation procedures that allow the user interface to adapt to the operator's needs.

3.1 System Architecture

In this thesis, a Physiologically Attentive User Interface (PAUI) was designed for application in the robot teleoperation field. For the purposes of this dissertation, the system was applied to a Search and Rescue robot developed from 2003 to 2005 called *RAPOSA* [25]. Nonetheless, the solution presented offers flexibility to be adjusted to other interfaces. The objective of the PAUI is to employ an artificial intelligent system trained to classify the operator's emotional state between three different emotional states (*Rest*, *Stress* and *Workload*), which is then used to update the interface dynamically with respect to the operator's needs. This allows the operator to remain focused on the robot teleoperation task, potentially leading to the achievement of a better performance on the mission. Furthermore, this user interface is developed over the pre-existing RAPOSA's user interface, which is a 15 years old interface with a complex display of information that can potentially overload the operator.

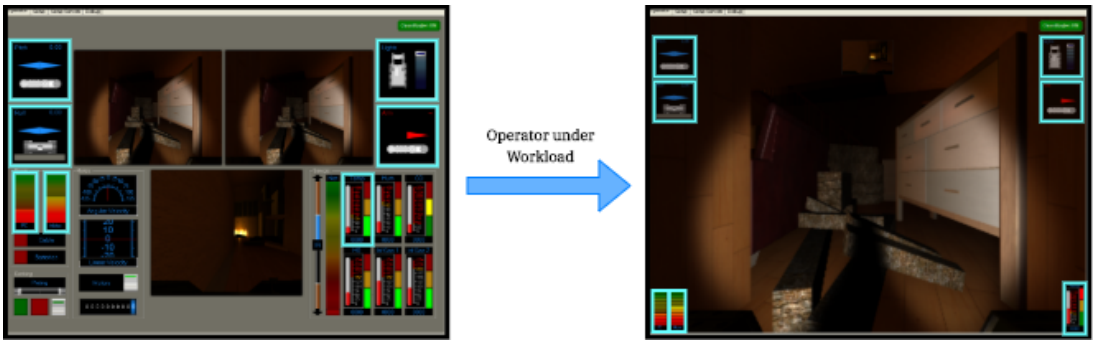


Figure 3.1: Simplified visualization of the PAUI concept. On the left, the complexity of the original interface is shown, with only the relevant elements of the interface highlighted in light blue. When the operator is under cognitive workload, the system updates the interface to a simplified version that aims to improve the operator's task performance, which is shown on the right.

The PAUI approach aims to reduce the complexity of this pre-existing interface in moments of cognitive stress, in order to ease the operator's workload based on his/her needs (see figure 3.1). This

approach also focuses on achieving this without access to the older user interface's source code. As such, an alternative method for building the new interface was developed, which brings together the extraction of the visual data of the older interface and the usage of a task automation tool that allows the PAUI to control the older interface, thus eliminating the need of creating a new interface or solution from the ground up and enabling the rearrangement of very old systems, such as RAPOSA, that are already established and can be hard to reprogram.

The creation of the PAUI relied on the combination of three different modules that worked together to make the concept of a PAUI applied in the robot teleoperation field feasible: the *Signal Extractor*, the *Emotional State Classifier* and the *Attentive User Interface*, which are going to be explained over the following sections. The combination of the three modules enables and communication between them allows the assembly of the PAUI. Figure 3.2 presents the overall architecture of the system:

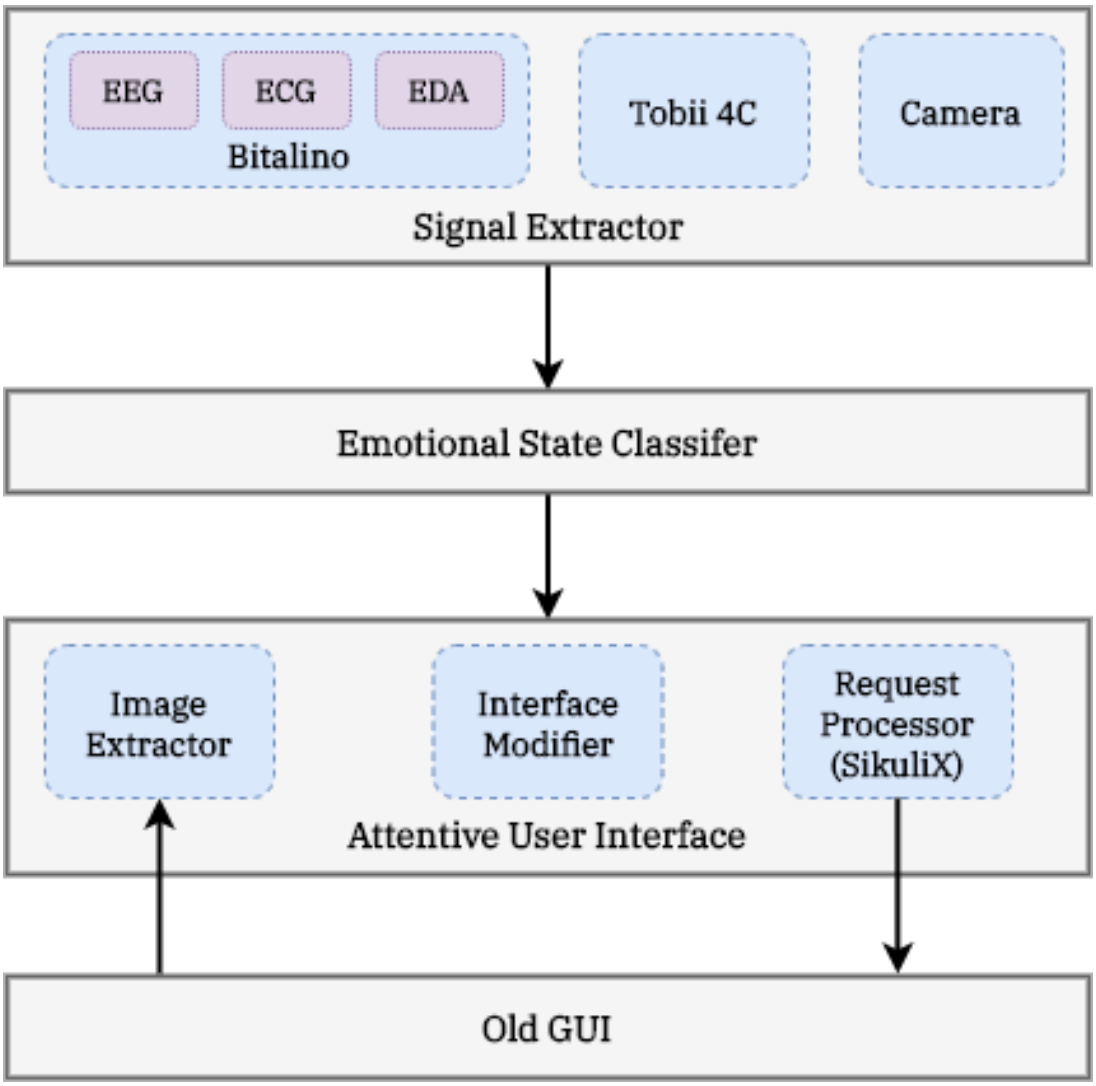


Figure 3.2: PAUI Architecture.

3.2 Signal Extractor

One of the core aspects of the system is the ability to estimate the operator's state of mind. In order to do so, the method requires the acquisition of data that can potentially yield valuable information about the emotional state of a person. Therefore, the extracted data comes from three main sources:

- Physiological signals
- Facial expressions and emotions
- Eye movements

With the purpose of collecting the data described above, different hardware devices were used. For reading physiological signals, the Bitalino (r)evolution Plugged Kit BT by Plux[1] was used to extract biosignals picked up by three sensors, namely EEG, EDA and ECG. For detecting eye movements, the device used was the Tobii Eye Tracker 4C by Tobii Technology[2]. Finally, for facial expressions and emotions, the laptop integrated webcam was used alongside the AFFDEX SDK by Affectiva [26].

The extracted data is then processed in order to obtain a wide range of parameters that can later be used as input for training a classifier. Data processing was performed with the aid of the pre-existing PAUI framework developed by Singh et al. (2018) [6], which processes the extracted raw analog signals into more refined parameters, such as heart rate from ECG, skin conductance level from EDA, brain wave frequencies from EEG facial expressions and emotions from the webcam, fixation information from the Tobii 4C Eye Tracker, amongst others. During execution, each device has a *thread* responsible for managing its data, which means three parallel running threads are needed to perform data acquisition and processing: the *Bitalino* thread which extracts physiological analog signals at 1000 Hz; the *Tobii* thread which monitors eye movements and fixation information at 90 Hz; and the *Camera* thread which extracts facial expressions and emotions at 30 Hz. A complete list of all the parameters that result from processing the extracted data can be found in table 3.1.

After extracting all the processed parameters, the Signal Extractor sends the signals to the emotional state classifier, which performs a prediction based on the data received.

3.3 Emotional State Classifier

Upon going through the data acquisition and processing step, the system can then feed the resulting parameters into a classifier that learns to identify the operator's emotional state. In this work, the data used to train the classifier was separated in three different classes: *Rest*, *Stress* and *Workload*. It should be noted that theses states were chosen as a basis for this work, as they target three possible

[1] Bitalino - (r)evolution Plugged Kit BT, Plux (Last accessed: 2020-06-02): https://bitalino.com/en/plugged-kit-bt
[2] Tobii Eye Tracker 4C, Tobii Technology (Last accessed: 2020-06-02): https://gaming.tobii.com/products

sensations felt by operators in teleoperation situations. There are more states that could be of interest, such as transitioning states between each of these, which can be explored in future works. In the previous work by Singh et al. (2018) [6], the classifier chosen to perform the emotional state prediction was a SVM, which obtained a classification accuracy of 88.37% for person specific data (classifiers trained individually for each person) and 84.75% for task specific data (classifier trained using data from all subjects). In this work, the classifier used was an *artificial neural network*, that was employed with the aim of improving the classification accuracy, considering DL techniques have been successful at solving problems in multiple areas of research in recent years (a thorough explanation of how the classifier was implemented and trained is given in chapter 4).

As referred in the previous section, the system runs a thread for each data extraction device that is in charge of acquiring and processing its respective signals. Likewise, the emotional state classification also has a thread responsible for its management, which runs above the three signal extraction threads. This thread receives the processed signals from the Bitalino, Tobii and Camera threads and makes an average of the signals received during 1 second. The thread then uses the averaged signals as input for the trained neural network, which predicts the operator's emotional state.

3.4 Attentive User Interface

Having gone through the prediction of the emotional state, the system can proceed to adapt the user interface to the operator's needs. The AUI module is responsible for managing the user's attention and ease the process of operating the robot in high cognitive stress situations. The AUI carries out this task by redefining the pre-existing interface, reducing its complexity and displaying only the relevant information at any given time. For this purpose, the AUI extracts image data (screenshots) from the old interface and uses it to render the new interface, which can be modified depending on the predicted emotional state. The AUI also issues requests to the old interface with the aid of *SikuliX*, a Java-based application that automates any task that can be done on the desktop screen using image recognition. The automation of mouse and keyboard operations allows the AUI to interact actively with the old interface. With the aim of hiding the old interface from the user's view while maintaining the interaction between both interfaces, the old interface runs on a virtual machine that is connected to the host machine, where the AUI operates. Chapter 5 gives a detailed description of the implementation procedures adopted to assemble the AUI. Following up, the strategies implemented by the AUI upon the detection of each emotional state are described. The AUI is the only module of the PAUI that needs to be tuned for each specific interface, since the procedures adopted for the management of user's attention are specific to the interface in question. As such, the procedures presented in sections 3.4.1, 3.4.2 and 3.4.3 are specific to RAPOSA.

3.4.1 Rest State

If the classifier predicts the user's emotional state as rest, the original interface is maintained. In a restful state, the operator can still manage his/her attention well enough to handle the complexity of the original interface and stay focused during the robot teleoperation task. As such, in this case, the AUI simply renders the extracted image of the original interface without modifying it. Figure 3.3 shows the appearance of the original RAPOSA's graphical interface.

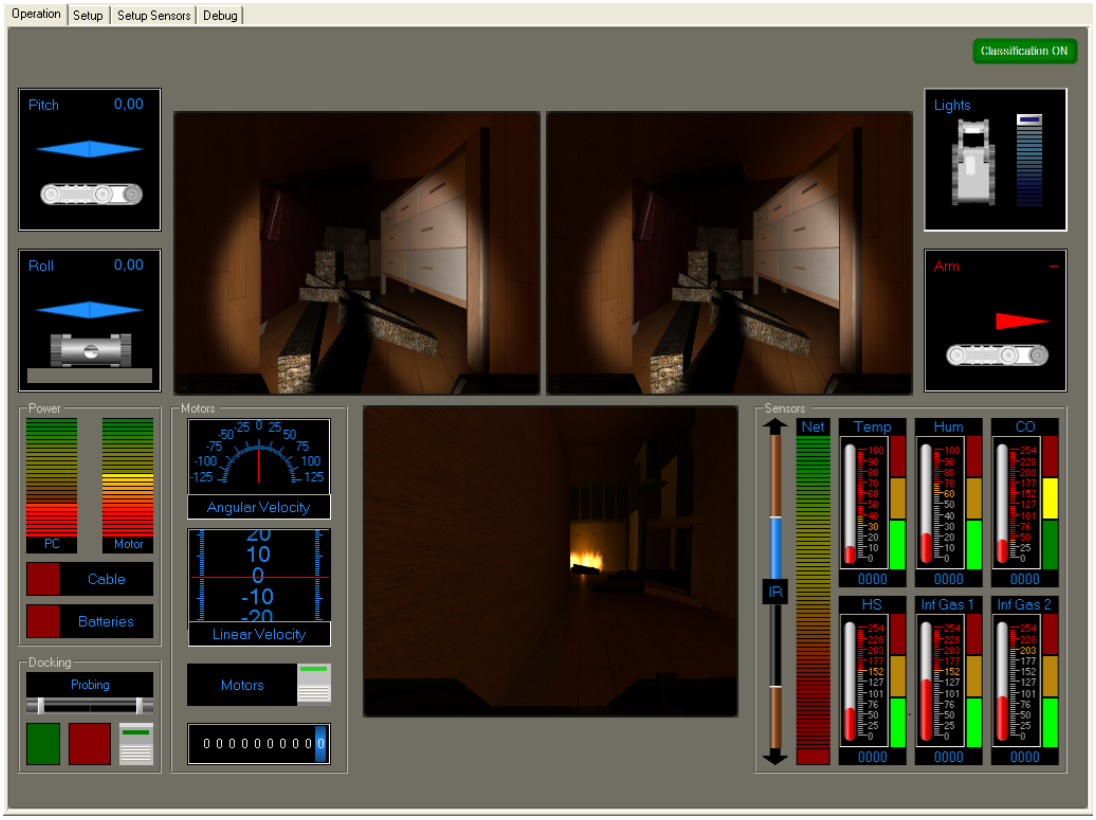


Figure 3.3: Original RAPOSA's GUI.

As can be seen, the original interface is quite complex. There are two frontal cameras side by side, one of which can be set to a thermal camera, along with a rear camera below the frontal cameras. In the upper left quadrant, two attitude indicators display the robot's pitch and roll angles. In the upper right quadrant there is a slider that shows the robot's spotlights intensity, as well as the robot's arm orientation. On the lower left quadrant, the interface presents both linear and angular velocity information, along with the battery levels for the motors and the electronics (PC). Finally, on the lower right quadrant there is a sensor panel that shows the values read for temperature, humidity, carbon monoxide, hydrogen sulfide and explosive gas levels.

3.4.2 Stress State

When the classifier predicts the emotional state of the operator as stress, the rendered interface remains to be the original GUI, but the AUI issues a request to the old GUI for an increase of the maximum speed of the robot. The Setup tab of the robot's interface provides a slider that allows the adjustment of the maximum speed of the robot. While its value could theoretically be set to the limit at all times, doing so leads to significant overheating of the robot at the hardware level. For this reason, the AUI takes up the responsibility of managing the robot's maximum speed, allowing it to reach values closer to the limit when the user is under stress, while alleviating the robot when the user does not need its full potential. This can be helpful in situations where some areas of the environment are clogged with difficult obstacles, which can cause the operator to lose precious time and enter a stressful state. For this reason, an increase in the maximum speed of the robot can help the operator compensate for these moments when the environment is easier to traverse, where he/she can make use of the extra speed without having to manually change it in the Setup tab of the interface. This attentive measure of the AUI relieves the user of having to adjust the robot's maximum speed whenever it is needed, avoiding unnecessary stress and additional delays in the mission. A clarification on how the AUI sends these requests to the old interface is presented in section 5.5. Also, it was chosen not to redefine the interface in this state since a stressful and anxious state is more likely to be caused by a difficult teleoperation environment rather than by the interface itself. As such, the user might benefit more from a raised maximum speed than an improved interface.

3.4.3 Workload State

Since operators usually focus mainly on the camera feed, most of the information shown in the interface is not relevant during the robot teleoperation task, or is only relevant beyond certain levels. For instance, the operator does not need information about the battery levels unless they are reaching a point that should cause concern. This idea is especially important in workload situations, where the operator is trying to execute a robot teleoperation task (e.g. scouting for victims in a ruined building), but has to divide his/her attention between performing the task and reading all the information the interface offers, e.g. checking the battery levels or making sure the temperature does not get too high when driving close to fires. The AUI takes advantage of this in order to manage the user's attention effectively, by redefining the interface to a simpler format that emphasizes the camera view and only shows information when it is needed. Figure 3.4 shows the new interface displayed in workload situations.

The new interface model is much simpler, reducing the amount of information presented drastically. Since the frontal camera is the main source of information for the operator, it has been stretched to fit the whole interface, with the rest of the interface elements sitting on top of it. Apart from the camera

image, the only elements that remain constantly on the interface are the attitude indicators, the lights slider, the arm angle indicator and the rear camera which was repositioned to the top of the interface in a smaller size, resembling a car's rear-view mirror. Also, some transparency was applied to these elements in order to make it possible to see through them (except on the rear camera). Additionally, all types of information that do not contribute directly to the performance of the task were removed, such as the linear and angular velocity and the docking panel. Furthermore, in order to manage user's attention effectively, the battery levels are only shown in the lower left corner when their values drop below 40%, and sensors are shown in the lower right corner individually when their values cross the danger zone. As the workload state is likely to be triggered when the interface is being a cause of distractions (e.g. sensor danger levels and low batteries), these measures contribute to maintaining the user's focus on the task, since the interface only demands more attention when it is absolutely essential.

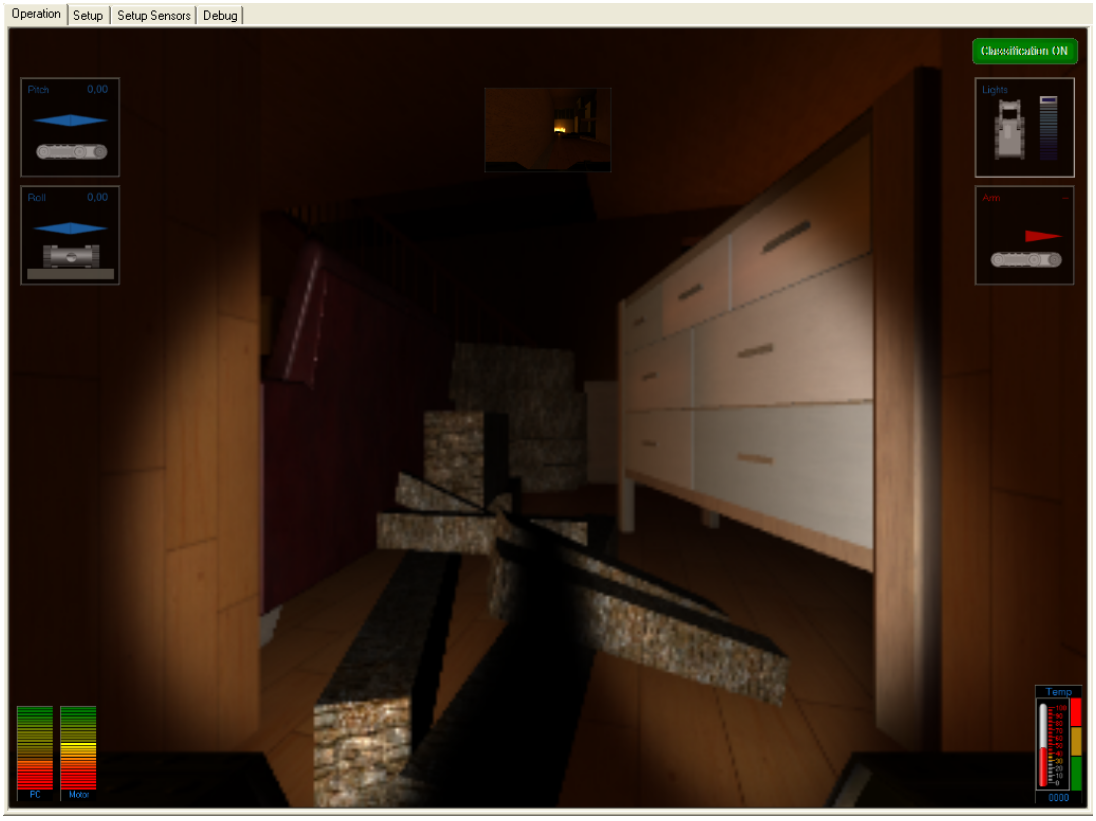


Figure 3.4: Redefined AUI rendered in workload situations.

Sensor	Parameters
ECG	• Heart Rate • Standard Deviation of Normal to Normal • Root Mean Square of the Successive Differences • Very Low Frequency (0.0033 Hz - 0.04 Hz) • Low Frequency (0.04 Hz - 0.15 Hz) • High Frequency (0.15 Hz - 0.4 Hz)
EEG	• Delta (0.5 Hz – 3.5 Hz) • Theta (3.5 Hz – 8 Hz) • Alpha (8 Hz – 13 Hz) • Beta (13 Hz – 30 Hz) • Gamma (30 Hz – 45 Hz) • Engagement (Beta / (Alpha + Theta)
EDA	• Skin Conductance Level • Skin Conductance Response
Tobii	• Number of Fixations • Total Time • Total Fixation Duration • Average Fixation Duration • Repeated Fixations • Biggest Fixation At • Maximum Visited Counts • Maximum Visited At
Camera (Emotions)	• Joy • Fear • Disgust • Sadness • Anger • Surprise • Contempt • Valence • Engagement
Camera (Expressions)	• Smile • Inner Brow Raise • Brow Raise • Brow Furrow • Nose Wrinkle • Upper Lip Raise • Lip Corner Depressor • Chin Raise • Lip Pucker • Lip Press • Lip Suck • Mouth Open • Cheek Raise • Dimplier • Eye Widen • Jaw Drop • Lip Tighten • Lip Stretch • Smirk • Eye Closure • Attention
Camera (Face Orientation)	• Pan • Tilt • Yaw

Table 3.1: List of processed parameters that result from the extracted signals.

4

Emotional State Classifier

Contents

In this chapter, the basic concepts behind the functioning of neural networks are explored, as well as how they are applied in this specific system. First, a brief mathematical introduction is given, regarding the most basic unit of a neural network, the neuron, as well as the basic functioning of the neural network's learning process. Afterwards, a general overview of the problem is given, determining what is the model's objective (i.e. what is it going to predict), as well as what data will be used as input for that prediction. The problem type in the context of DL is then presented, establishing a metric to measure the success of the model, as well as a validation protocol. Subsequently, data preparation procedures are discussed and the network architecture design is explained. Finally, the problem of overfitting is discussed, as well as how regularization techniques can be applied to fight it, along with procedures used for visualizing a dataset and reducing its dimensionality.

4.1 Mathematical Introduction and Concepts

In order to understand how neural networks work, the functioning of the essential cell of a network, the neuron, must be acknowledged. Although many types of artificial neurons are used, the *sigmoid* neuron is explained here, as it paves the way for more complex types of neurons. Figure 4.1 shows the schematic of a sigmoid neuron.

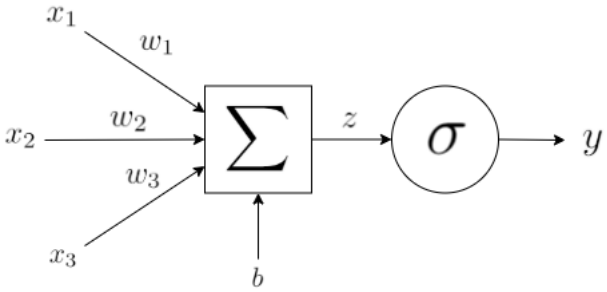


Figure 4.1: Schematic of a sigmoid neuron.

In this example, the neuron receives three inputs and produces one output, but the number of inputs can be variable. The output of a neuron is given by:

$$y = f\left(\sum_i w_i x_i + b\right) \tag{4.1}$$

In equation (4.1), x_i represents the input of the neuron, w_i represents the weights given to each input and b is the bias of the neuron. The function f represents what is called the *activation function*, which, in the case of the sigmoid neuron, is the sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{4.2}$$

There are other choices of activation functions apart from the sigmoid function, which will be analysed later in section 6.7.1. For now, the sigmoid function will serve as an example for the rest of the explanation. As can be seen, a change in one of the inputs of a neuron will cause a change in the output, which can be more or less significant depending on the weight attributed to each specific input. The variation of the weights and the bias of a neuron yields different models of decision-making [27]. The connection of multiple neurons is called a *neural network*. Figure 4.2 represents a basic neural network diagram.

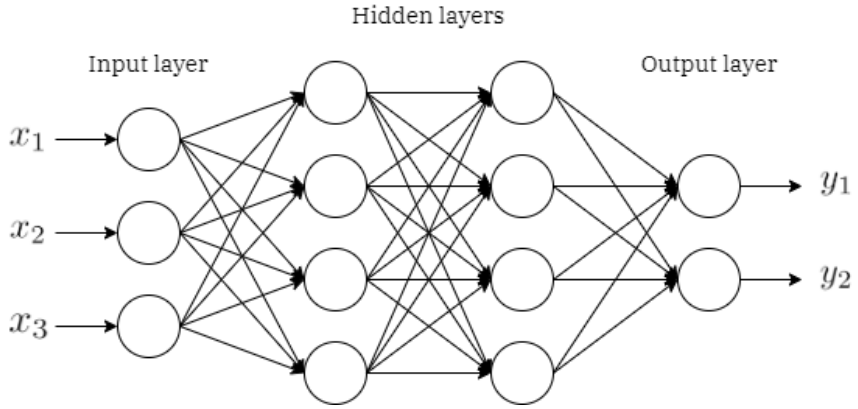


Figure 4.2: Diagram of an neural network example.

Neural networks are often organized in layers of neurons, which gives the network the ability to represent more complex information as the network gets deeper (i.e. as the number of layers and neurons increases). In the example above, the terminology used for naming the layers and the respective neurons is given. The leftmost layer is referred to as the *input layer*, the rightmost layer is the *output layer* and the inner layers are called *hidden layers*. In this example, the network portrayed is a *feedforward* neural network, since the neurons of each layers are connected to all the neurons in the next layer, which means information is always travelling forward in the network.

After understanding the basic functioning of a neural network, the learning process can be resumed to the minimization of a *cost function*, most frequently called *loss function*. The choice of loss function depends heavily on the problem, but it usually gives a measure of how close the output of the network is to what it should be, based on the weights and biases of the network. Having the loss function defined, the *gradient descent* algorithm is then used to find the weights and biases that lead to a minimum of this function. If we designate the loss function as $J(w, b)$, w being the weights and b the biases, the gradient descent algorithm can then be summarized in the following two equations:

$$w = w' - \alpha \frac{\partial J(w,b)}{\partial w} \tag{4.3}$$

$$b = b' - \alpha \frac{\partial J(w,b)}{\partial b} \tag{4.4}$$

In equations (4.3) and (4.4), the derivative term represents a general direction of movement towards a minimum, while α is the *learning rate*, which defines how fast the algorithm will approach the minimum. The weight and bias updates are repeated through multiple iterations until gradient descent finds a minimum to the loss function. However, this can be very computationally expensive, since the loss function is actually an average over the costs for each individual training sample. This means that the model has to compute the gradient for all training samples in the dataset just for one single gradient descent step. As such, in practice, it is much more efficient to use a variation of this algorithm called *stochastic gradient descent*, which takes a small set of random training samples to compute each step, instead of using the whole dataset [28]. This set of samples is called a *mini-batch*. The size of the batch represents a compromise between speed of computation and precision of the gradient average, since smaller batches will produce less accurate gradient averages. The algorithm then performs a gradient descent step for each random mini-batch until the whole training dataset has been seen, thus completing an *epoch* of training. Deep learning models are usually trained over multiple epochs shuffling the data between each epoch in order to generate new batches of data each time.

The only step left in applying this learning procedure is being able to compute the loss function gradients with respect to each weight and bias. For this, the *backpropagation* algorithm is used, which propagates the error of the predicted output versus the expected output backwards through the network, updating the weights and biases each time. According to Nielsen (2015) [27], this algorithm comprises four main equations:

$$\delta^L = \nabla_a J \odot \sigma'(z^L) \tag{4.5}$$

$$\delta^l = ((w^{l+1})^T \delta^{l+1}) \odot \sigma'(z^l) \tag{4.6}$$

$$\frac{\partial J}{\partial b^l_j} = \delta^l_j \tag{4.7}$$

$$\frac{\partial J}{\partial w^l_{jk}} = a^{l-1}_k \delta^l_j \tag{4.8}$$

In these equations, δ^l_j represents the error for neuron j in layer l, z^l represents what is called the *weighted input* of the neurons in layer l (i.e. the output of the neuron before the activation function is applied), a^l_k is the output of neuron k in layer l (also called *activation*), L represents the output layer, b^l_j is the bias of neuron j in layer l and w^l_{jk} is the weight from the neuron k in layer $l-1$ to the neuron j in layer l. The absence of a subscript in these values represents the respective element in matrix-form (e.g. δ^L defines the error vector in layer l). These equations allow the computation of the error in all layers, as well as the respective gradient of the loss function with respect to each weight and bias, thus making the stochastic gradient descent calculations feasible. The updates performed to the weights and biases symbolize the neural network's learning process.

4.2 Problem Definition

Now that the base concepts behind a neural network are explained, the problem can be defined in the DL setting. In order to do so, the input data and the prediction objective must be specified. As referred before in chapter 3, the classifier's goal is to predict the user's emotional state from three different classes: *Rest*, *Stress* and *Workload*. The input data used to carry out this prediction is also presented in the same chapter, in table 3.1. Using this information, the problem type can be picked out of many possible problems that are solvable by neural networks, such as *regression*, *classification*, *clustering*, *generation*, amongst others [29]. Since the model's objective is to identify the emotional state label that each observation fits in, the problem falls in the *classification* field. Depending on the number of classes specified, the problem can either be a *binary classification* problem (two classes) or a *multiclass classification* problem (three or more classes). As such, this problem belongs to the latter type. Within the multiclass classification area, a problem is said to be a *multi-label classification* problem if each instance can be categorized in multiple classes simultaneously, or it can be a *single-label classification* problem if each instance can only fit a single class, which is the case here since a person can not be in rest and stress at the same time, for example. Summarizing, the problem of classifying a person's emotional state as rest, stress or workload is defined as a *single-label multiclass classification* problem.

4.3 Metrics and Validation

Having the problem defined, a quantitative measure of the model's performance can be established. Normally, this measure is used to evaluate the model's predictions in a *test set*, which is set of observations the model has never seen before. This set is separate from the *training set*, which is the set of points the network uses to train its parameters [30] (dataset split procedures will be further explained in section 4.4.2) . The choice of this measure is heavily dependent on the task and on the type of problem at hands. In order to facilitate the comprehension of each metric, one should first understand the concept of a *confusion matrix*. This matrix is a representation of the classes predicted by a model against real labels. From this matrix, the four following concepts arise:

- **True Positive (TP):** A positive detection that is predicted as positive;
- **False Positive (FP):** A negative detection that is predicted as positive;
- **True Negative (TN):** A negative detection that is predicted as negative;
- **False Negative (FN):** A positive detection that is predicted as negative.

These concepts are defined in the context of a binary classification problem, where only two classes are trying to be predicted, but it is easily extended to a multiclass classification problem by simply

considering a binary problem for each class versus all the others.

According to Goodfellow et al. (2016) [30], some of the most common metrics for classification tasks, are *accuracy*, *precision*, *recall* and *F1-score*. A description of each of these metrics is displayed below in table 4.1:

Metric	Description	Expression
Accuracy	Measures the ratio between the samples correctly classified by model and the total number of samples	$\frac{TP+TN}{TP+TN+FP+FN}$
Precision	Measures the fraction of events predicted by the model as being true that were indeed correct	$p = \frac{TP}{TP+FP}$
Recall	Measures the fraction of true events that were predicted as being true	$r = \frac{TP}{TP+FN}$
F1-score	Measures the overall accuracy of the model as a combination of precision and recall	$\frac{2pr}{p+r}$

Table 4.1: Description of common metrics used to measure model performance in classification problems.

One important aspect to consider in the choice of a metric is the dataset class balance. While accuracy can be a simple and effective way to measure a model's performance, it may not be the most suitable metric for class-imbalanced problems. For instance, if a dataset contains 90% of observations belonging to class A and only 10% belonging to class B and the model is hard-coded to predict all observations as class A it will obtain an accuracy of 90%, even though it did not learn anything about the dataset itself. In such cases, it might be better to use precision or recall as a metric, which in cases of multiclass problems are applied to each class. However, as will be seen later in section 6.5, the dataset collected for the purposes of this thesis is well-balanced, and as such, the chosen metric for measuring the neural network's performance was accuracy.

Another step necessary in preparing the model for training is to decide how to validate the model. In neural networks training it is common to have a set of observations that is used to tune the parameters of the network and see if overfitting is taking place (overfitting is explained further in section 4.6). This set is called the *validation set*. Normally, the validation protocol used is *hold-out validation*, where a small section of the training set is taken to be used for validation. This protocol works well for large datasets, where even a small section should be representative enough of the underlying distribution of the data. However, for small datasets, such as the one that is used in this work, the validation split could lead to high variance in the validation scores. Therefore, the validation protocol chosen for this work was *K-fold cross validation*, which instead splits the training set into K partitions and trains K models, where each model uses a different partition of the training set as its validation set. The final validation score

is computed from the average of the validation scores of each model. After tuning the network, a final model is trained on the entire training set, using the best parameters and its performance is verified on the test set. The value of K is usually defined as a compromise between the percentage of data the model is trained on and the fidelity of the generalization error, since larger values of K lead to smaller folds, which means that each validation set might not be representative of the data, thus leading to higher variance. In problems where a large amount of data is available, larger values of K may be a viable option, since each fold will still contain enough data to represent the data distribution accurately. However, in this case, the dataset acquired is small, so a smaller value of K is preferred. In this work, the recommendations of Anguita et al. (2012) were followed, which state that the optimal value of K lies between 3 and 4 for problems where the reliability of the generalization error is of interest [31]. As such, a value of $K = 4$ was adopted.

4.4 Data Preparation

Before inputting the data samples in the network, it must go through a preprocessing step in order to make the data more tractable for the model. Naturally, neural networks must receive their input data in the form of vectors of numbers, either floating-point data or integer data, so they can perform calculations with these numbers. In this case, as will be seen in section 6.7.1, each sample in the dataset is made of 45 features, which means that each input sample is a 45-dimensional vector. Fortunately, the features extracted are all floating-point numbers, so the data is already vectorized and in a format that the neural network can digest.

4.4.1 Dataset Organization

Although data collection procedures will be discussed in detail in section 6.5, a brief contextualization of the dataset organization is presented in this section, with the intent of aiding the comprehension of the dataset splitting process.

As stated before, the observations collected are labelled in three classes: rest, stress and workload. In order to collect data for each of these classes, a task designed to induce each of these states was performed. As such, the dataset will be composed of three different timelines, one for each class, where data was collected over the course of time of the task for each class. The data from the three tasks was then simply concatenated in order to create the entire dataset. As such, the raw dataset is organized by class, and the data from each class is ordered by the timestamp of collection, thus constituting a sequence of discrete-time data. Below is a diagram that summarizes the organization of the dataset:

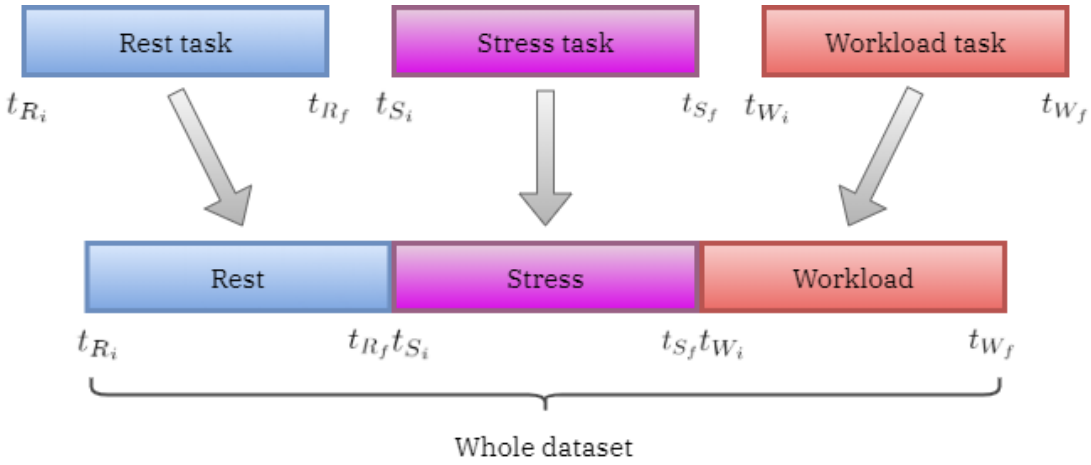


Figure 4.3: Dataset organization diagram.

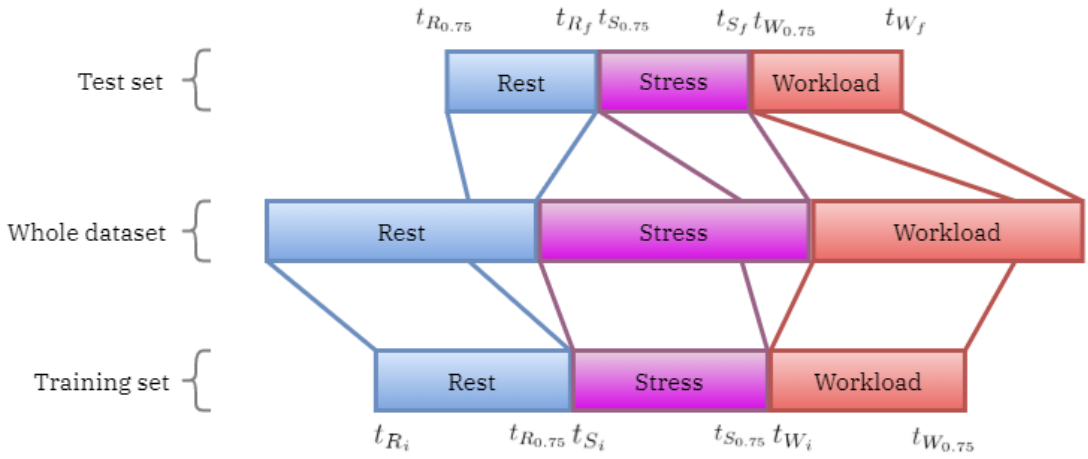
In figure 4.3, t represents the timestamps, where each subscript R, S or W specifies the task to which the timestamps belong (Rest, Stress and Workload respectively) and the subscripts i and f indicate whether it refers to the initial or the final timestamp of the task.

4.4.2 Splitting the dataset

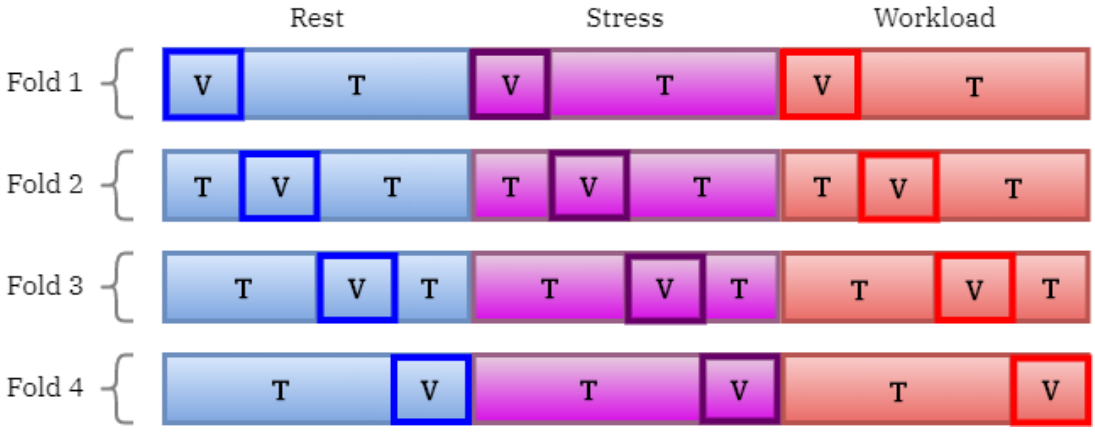
After loading the dataset, all the available data needs to be split three ways into a training set, a validation set and a test set. As the names suggest, the training set is used to train the neural network, which is then evaluated using the validation set. The reason the model is evaluated on a validation set and not on the test set directly is due to the fact that building a model requires tuning *hyper-parameters* such as number of hidden layers, number of hidden neurons, learning rate, batch size, amongst others. After tuning the model's configuration, the model is tested one last time using the test set, which is never exposed to the network before, not even indirectly. This way, the performance on the test set gives a measure of how the model will perform in real situations. The absence of a validation set would compromise the legitimacy of the test set, since every time a hyper-parameter is tuned based on the validation scores, some information about the validation set is leaked to the model. When this process is repeated multiple times, the model will essentially be optimized for the validation set. If the test set was used instead, then the performance on the test set would not directly reflect what would happen in real-life.

Normally, before proceeding to split the dataset, it is randomly shuffled in order to guarantee that both the training set and the test set are statistically representative of the data distribution. However, as was discussed in section 4.4.1, the data collected for the purposes of this thesis is time-series data, where the user's biometric and facial expression signals were collected over the course of time for each class. As such, shuffling data before splitting would mix observations from different timestamps, and the network

training process would expose it to the whole array of time, which would probably lead to misleading good results on the test set [29]. Therefore, the data was split without shuffling in a 75/25 ratio, using the first 75% of each class for the training set and the remaining 25% for the test set. This ratio represents a compromise between having more variance on the parameter estimates (which increases as the training set size decreases) and having more variance in the performance measure (which increases as the test set size decreases). Since the validation protocol used in this work is K-fold cross validation with $K = 4$, the training set is then split 4 ways as referred in section 4.3, using each fold as a validation set to train a different model. The validation set split is done in a similar fashion to the train-test split, where each fold i generates a validation set that contains data from the i^{th} part of each class. A visual clarification of the train-test split and the validation split procedure is shown below in figure 4.4:



(a) Train-test split. The subscript 0.75 indicates the point of separation between training and validation data. This point is included in the validation set and excluded in the training set.



(b) Validation split. V represents the portions that are included in the validation set and T indicates the portions that are included in the training set. The sections extracted from each class are then concatenated in order to construct the whole training and validation sets.

Figure 4.4: Visualization of the dataset splitting procedures.

After doing the validation split for each fold, the remaining training data is then shuffled before proceeding to train the network. This step is very important, since stochastic gradient descent uses batches of the training set to compute the gradient towards a minimum of the loss function. Since the data is organized by class, skipping the shuffling step would mean that batches would most likely contain data from a single class, which means that each batch would not be representative of the overall distribution, thus preventing the computed gradient from being close to the ”true” gradient.

4.4.3 Normalization

In addition to preprocessing and splitting the dataset, another common measure to ensure the neural network converges easily is to normalize the samples before feeding them to the model. Data normalization is especially important when features take up large values and are in different ranges from each other. When this is the case, a small change in the value of one of the features can lead to a significantly different change in the output when compared to the same change in other feature. By normalizing each feature to have a mean of 0 and a standard deviation of 1, the model assigns the same importance to each feature, leading to faster convergence to a local or global minimum. The features present in the dataset used for this work take up values in significantly different ranges, and as such, data normalization was applied. Depicting the dataset as a matrix X, where each line i represents a sample and each column j represents a feature, normalization can be applied as follows:

$$X_{norm_{i,j}} = \frac{X_{i,j} - mean(X_j)}{std(X_j)} \tag{4.9}$$

It should be noted that the mean and standard deviation used to normalize the validation and test sets are computed using the training set. This measure prevents the leakage of information about the validation and test sets to the model during training.

4.5 Model Configuration

Before creating a model that can effectively train on the prepared data, a loss function that will guide the model's training must be chosen. Since the problem at hands is a multiclass classification problem, the loss function chosen in this work was the widely used *categorical cross-entropy* loss function [32], which gives a measure of dissimilarity between the predicted label and the true label:

$$J(y, \hat{y}) = -\sum_{i=1}^{C} y_i \log \hat{y}_i \tag{4.10}$$

Here, C represents the total number of classes, y is the target label vector, which is a one-hot

encoded C-dimensional vector where all entries are zero except the target label entry which is one, and $\hat{y}$ is the output vector predicted by the model. The $\hat{y}$ vector is calculated using the *softmax* function, which was used as the activation function of the output layer of the model in this work, as is common in multiclass classification problems [29]. The softmax function takes the weighted input vector z of the last layer of the network and normalizes it into a vector where each entry i is the estimated probability that the sample belongs to class j given the scores of each class for that same sample:

$$\mathcal{S}(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{4.11}$$

While there is not a direct relationship between the accuracy of classification, which is used as a metric of the performance of the model, and the value of the cross-entropy function, it is generally expected that as the predicted values get closer to the true labels the accuracy of the model will rise.

Having established a loss function, there is also the need to choose an optimizer to lead it to a minimum. Although the most basic optimizer is gradient descent, more recent optimization strategies lead to a faster convergence of the loss function. Most modern optimizers use gradient descent as a basis, but work around its flaws. In this work, a state-of-the-art optimizer called *Adam* was used, which is an efficient optimization method that has low memory requirements and has been shown to be advantageous in practice for most problems, when compared to other stochastic optimization methods, such as *Nesterov accelerated gradient*, *AdaGrad* and *RMSProp* [33]. The default initialization values for all the hyper-parameters except the learning rate were used: 0.9 for the momentum decay hyper-parameter (β_1), 0.999 for the scaling decay hyper-parameter (β_2) and 10^{-8} for the smoothing term (ϵ) [34]. Some values were tested for the initial learning rate of the optimizer which are going to be analysed in section 6.7.1.

4.6 Overfitting and Regularization

Over the course of training a DL model it is very common to witness a phenomenon called *overfitting*. Overfitting refers to the point where the performance of the model on the validation data starts degrading over the epochs of training. As stated before, the goal of a model is to achieve *generalization*, i.e. the ability to perform well on data that has never been seen before. However, as seen before, the training of a model constitutes an *optimization* problem, where the best performance possible on the training set is achieved. In the first epochs of training, the network is learning the basic patterns present in the training data, that are also present in the validation data, and thus, the performance on both sets usually keeps increasing until it reaches a point where the performance on the training set keeps increasing but the validation score starts decreasing. At this point, the network is no longer learning to recognize the essential patterns in the data, but is instead “memorizing” all the peculiarities of the training set, which

are irrelevant in classifying new samples of data. The model is said to be *overfit* to the training data at this point.

There are many possible strategies to combat overfitting, most of which involve limiting the potential of the network to store information in its weights and biases. The process of applying these measures in order to help the model generalize better is referred to as *regularization* [35]. The most common regularization methods include:

1. **Reducing the network's *capacity***: The capacity of a network refers to the function space that it is able to represent as its weights and biases vary [36]. The model capacity is usually related to the number of hidden layers and hidden neurons present in the architecture. As the number of layers and neurons increases, so does the capacity, which means that larger models are able to represent more complex functions and potentially solve more difficult problems. While this may be appealing, creating a model that is too complex can easily lead to overfitting, as the patterns present in the data are much simpler than what the model is trying to recognize, which leads it to perceive patterns that do not exist. By adjusting the model's capacity to the problem, overfitting is less likely to occur, as the complexity of the model matches the complexity of the problem.

2. **Dropout:** *Dropout* is a technique where the output of a random subset of neurons is set to zero, as well as their connections with other neurons. By dropping out random neurons, noise is introduced in the model, which can effectively disintegrate insignificant patterns captured by the model [37].

3. **Weight regularization:** Another way to mitigate the risk of overfitting is to force the network to give small values to its weights. In the same way that reducing a model's capacity decreases the risk of overfitting, smaller weights also lead to a simpler model, which is less likely to overfit. The way to do this is to penalize the model by introducing a cost in the loss function related to having large weights [30]:

$$\hat{J}(w) = J(w) + \lambda\Omega(w) \tag{4.12}$$

Here, $\hat{J}(w)$ denotes the regularized cost function, $\Omega(w)$ is a cost associated with the weights of the model and λ is a hyper-parameter that sets the amount of regularization to be applied. Larger values of λ lead to more regularization. One of the most common parameter penalties applied is known as L^2 *regularization*, where $\Omega(w) = \frac{1}{2}|w|^2$. This procedure is also know as *weight decay*. By introducing a cost on the square of the weight values, only the parameters that contribute significantly to the minimization of the cost function are preserved, whereas weights that might not be so informative tend to decay to smaller values, thus increasing the generalization power of the model. Another possible parameter penalty is known as L^1 *regularization*, where $\Omega(w) = |w|$. Here, the cost applied is proportional to the absolute value of the weights, which leads some weights to shrink to values very close to zero. Therefore, L^1 regularization leads to a sparser

solution, since some features will effectively be removed, making this approach act as a feature selection tool.

4. **Early stopping:** A simple way to avoid overfitting is known as *early stopping*. As the name suggests, it consists simply of stopping the training process when the model performance on the validation set stops improving.

The process of tuning hyper-parameters, defining the model's architecture and choosing regularization methods relies on observing the behaviour of the loss and accuracy of the model during each epoch of training, both for the training set and the validation set. As the objective is to find the ideal architecture of the model that stands on the border between overfitting and underfitting, the best approach to arrive at this point is to start with a small model and progressively add hidden layers and neurons until the model starts overfitting [29]. At this point, regularization techniques can be applied in order to obtain a regularized model that neither overfits nor underfits. The neural network architecture (i.e. the number of hidden layers and hidden neurons, as well as the activation functions of each layer), along with the choice of regularization methods and the results obtained for each of them are explored further in section 6.7.1.

4.7 Dimensionality Reduction

When dealing with high-dimensional feature spaces, such as the present one, it is common to apply dimensionality reduction techniques, not only as a measure to prevent overfitting, since feature spaces with lower dimensions tend to overfit less, but also to improve the prediction accuracy of the models. Most techniques for dimensionality reduction either fall on the *feature elimination* category or the *feature extraction* category. Feature elimination relies on completely dropping some features from the dataset that are known to have lesser relevance than other for the classification task. While feature elimination has the advantage of preserving the interpretability of features, dropping some features can lead to the loss of any benefit those features might bring. In this work, since there is little relevance in maintaining the interpretability of the features, a feature extraction method was adopted in order to retain valuable information from all features. For this purpose, the feature extraction method explored was PCA, a well-established technique which essentially projects the existing dataset in a new feature space that represents the variation of features in the best way possible [38]. In order to understand how PCA accomplishes this, let $\mathbf{X}$ be a $n \times m$ data matrix, where each line n represents a different observation and each column m represents a feature of the dataset that has been centered (i.e. its sample mean has been shifted to zero). The $m \times m$ covariance matrix $\mathbf{C}$ can be computed from $\mathbf{X}$ by:

$$\mathbf{C} = \frac{1}{n}\mathbf{X}^T\mathbf{X} \tag{4.13}$$

Considering $\mathbf{V}$ as a $m \times m$ matrix where each column represents an *eigenvector* of $\mathbf{C}$ and $\mathbf{D}$ as a diagonal matrix where the elements of the diagonal represent the *eigenvalues* of $\mathbf{C}$, the covariance matrix $\mathbf{C}$ can be factorized as:

$$\mathbf{C} = \mathbf{V}\mathbf{D}\mathbf{V}^{-1} \tag{4.14}$$

By sorting the columns of $\mathbf{V}$ and the eigenvalue matrix $\mathbf{D}$ in order of decreasing eigenvalue, the first p columns of $\mathbf{V}$ can be maintained as the $m \times p$ matrix $\mathbf{T}$. The $n \times p$ data matrix $\mathbf{Y}$ projected by PCA in the new feature space is then given by:

$$\mathbf{Y} = \mathbf{X} \cdot \mathbf{T} \tag{4.15}$$

In sum, PCA calculates the covariance matrix of the dataset, along with its eigenvalues and eigenvectors, where the covariance matrix gives a measure of how correlated features are between each other, the eigenvectors represent the directions in which the data is dispersed (also called *principal components*) and the eigenvalues represent the importance of each of these directions. By projecting the dataset in this new set of directions, the dimensionality can be reduced by dropping the principal components whose eigenvalues are lower, i.e. explain less variability in the data.

Apart from the reduction of overfitting tendencies and the propensity to obtain better classification accuracies, dimensionality reduction can also be useful in data exploration and visualization. One such technique that is particularly well-suited for the visualization of high-dimensional datasets is t-distributed Stochastic Neighbour Embedding (t-SNE). This method embeds a high-dimensional dataset in a low-dimensional space (2 or 3 dimensions) by measuring similarities between points x_i and x_j, both in the high-dimensional space and the low-dimensional space, and converting those similarities into joint probability distributions P and Q. These distributions represent the probability p_{ij} or q_{ij} of each point i picking another point j as its neighbour, given the similarity between those points, for both the high-dimensional space (p_{ij}) and the low-dimensional space (q_{ij}) [39]. Through the minimization of the Kullback-Leibler divergence between both distributions with regard to a defined cost function, the low-dimensional embedding achieves a faithful representation of the high-dimensional dataset:

$$KL(P \parallel Q) = \sum_i p_{ij} \log \frac{p_{ij}}{q_{ij}} \tag{4.16}$$

The application of a t-SNE requires the definition of two primary hyper-parameters: perplexity and number of iterations. Perplexity is interpreted as a measure of the number of neighbours of each data point, which means that larger datasets require higher values of perplexity. Its value is typically set between 5 and 50, although the algorithm can be robust to changes in the perplexity [39]. Regarding the number of iterations, its value should be increased up to the point where the convergence stabilizes, although very high values can also be computationally expensive.

5

Attentive User Interface

Contents

In this chapter, the main functionalities of the AUI are discussed. First, an overview of the development libraries that were used in the development of the new interface are presented. Then, the process of extracting images from the old GUI without access to its source code is explained. Afterwards, the communication protocol that allowed the AUI to access the old GUI data is explained, along with the reasoning behind the choice of protocol. Subsequently, the implementation procedures adopted to modify the interface with respect to the operator's emotional state are addressed, taking place in the context of robot teleoperation. Finally, the processing of requests issued by the AUI and the execution of the required activities in the old GUI are examined, as well as the strategies implemented to speed up the processing of these requests.

5.1 Development Libraries and Tools

Before proceeding to explain the capabilities of the new interface, a review of the development libraries that aided in the design of the AUI is presented in this section. The programming language used in the development of this software was the most recent version of the C++ standard, *C++17*[1]. One of the main reasons C++ was used in this work was to continue the development of the previous work by Singh et al. (2018), which was implemented in C++. Nevertheless, C++ is an object-oriented language that provides a level of abstraction with little to no overhead at run-time. This is ideal for projects where performance in real time is very important and can have a significant impact on the execution of tasks, like in the robot teleoperation field, particularly in USAR operations such as the one at hands.

Apart from the C++ standard libraries, other external libraries were used, such as the *Boost C++ Libraries*[2] (version 1.63.0), which was used mainly for the implementation of the networking protocol. As the solution also requires manipulation of windows, the *Win32 API*[3] was used in order to make the connection between the old GUI and the AUI possible. Also, in order to create the new interface, a graphics library had to be used in order to display the data from the old interface and apply changes to it. To make this possible, the *SDL* (Simple DirectMedia Layer)[4] library was used (version 2.0.12). Finally, in order to fill the communication gap between the old GUI and the AUI, a task automation tool called *SikuliX*[5] (version 2.0.4) was employed.

[1] ISO/IEC. (2017). ISO International Standard ISO/IEC 14882:2017(E) – Programming Language C++. Geneva, Switzerland: International Organization for Standardization (ISO)
[2] Boost C++ Libraries, Boost (Last accessed: 2020-06-25): http://www.boost.org/
[3] Windows API, Microsoft Corporation (Last accessed: 2020-06-27): https://docs.microsoft.com/en-us/windows/win32/
[4] Simple DirectMedia Layer, SDL (Last accessed: 2020-06-28): https://www.libsdl.org/
[5] SikuliX by RaiMan (Last accessed: 2020-06-28): http://sikulix.com/

5.2 Interface Image Extraction

One the main goals of this solution is to be able to perform changes to a pre-existing graphical interface in real-time, based on the user's emotional state, without access to that interface's source code. This limitation brings the need for an approach that focuses on extracting visual data from the older GUI and using it to create a new interface based on the older one. As such, the solution found to this problem was to take screenshots of the old interface window and transmit them in a reliable and continuous manner to the new interface, which in turn displays the screenshots in a new window that constitutes the AUI. The old GUI window is hidden, meaning the user only interacts with the AUI.

The first step in enabling this solution is to take screenshots of the old interface window. In this work, a C++ class called *InterfaceScreen* was created for the purpose of handling all the necessary operations involved in taking screenshots of the old GUI window. This class makes use of the Win32 API in order to take a screenshot of any window of choice in a Windows operating system, where each instantiated object takes the title of the desired window as a parameter (the title is passed to the constructor of the class as a *string* type object). This means that the class can be used to take screenshots of any other interface of choice to be applied in other systems. When an *InterfaceScreen* object is instantiated, the class calls a private method called *GetInterfaceHandle*, which searches all currently open windows and finds the one which name corresponds to the one received, and then proceeds to store the *handle* to that window. A handle is a type of object defined in the Win32 API that acts as an identifier of the window, providing abstraction of the internal Win32 resources from the API user. If the method fails to find the required window (e.g. if it is not open), it returns an error value and the program shuts down.

After the handle to the old interface's window is retrieved, the class calls an initialization method that creates a *device context* (DC) compatible with the device context of the window. According to the Win32 API documentation[6], a device context is a "structure that defines a set of graphic objects and their associated attributes". The method then proceeds to store the dimensions of the window and create a *BITMAP* object that will hold the captured image. Finally, the method allocates memory for a byte array with the size necessary to hold the screenshot data, based on its dimensions, and initializes a *BITMAPFILEHEADER* structure, which holds information about the type, size and layout of the bitmap, and a *BITMAPINFOHEADER* structure that contains the dimensions and color format of the image.

Following up on the initialization of the class, another method called *takeScreenshot* can be called at any time to take a screenshot of the old interface window. This method calls the Win32 API function *BitBlt*, that executes a bit transfer of the *RGB* data of the specified rectangle's pixels (in this case, the old interface's window) to the BITMAP object, followed by the calling of the function *GetDIBits*, which stores the bits in a buffer. Therefore, the screenshot is held in memory and can later be sent to the AUI, which will use the image data from the old GUI to construct the new interface.

[6] Device Contexts, Win32 API (Last accessed: 2020-06-30): https://docs.microsoft.com/en-us/windows/win32/gdi/device-contexts

5.3 Communication Protocol

The next step in the creation of the new interface is establishing a connection between the old GUI and the AUI that allows a continuous screenshot transfer, essentially constituting a *image streaming* service. In this work, since the robot used (RAPOSA) was very old, its original interface was developed for usage in Windows XP, and was not compatible with more modern operating systems. As such, the physical machine that held the software for RAPOSA's interface was converted in a virtual machine using VMware vCenter Converter Standalone, for use in VMware Workstation[7] 10.0.7. Another reason that motivated the use of a virtual machine was the necessity of hiding the original interface from the user's view, while retaining the communication between both interfaces. Thus, the original interface was running in the virtual machine while the new interface was running on the host machine, meaning the communication protocol was established between these two machines.

A communication protocol is a set of rules that guides the transmission of data between two devices [40]. The two most common protocols in usage are part of the *Internet Protocol* suite: the Transmission Control Protocol (TCP) and the User Datagram Protocol (UDP). Both of these protocols have different ways of handling the transmission of *packets* (units of data) over a network. While TCP acknowledges the establishment of a connection and transmits packets in an ordered and reliable manner, closing the connection after all data was transmitted, UDP is a simpler protocol that does not need to establish a connection and does not guarantee data integrity. The main differences between TCP and UDP can be found in table 5.1:

	TCP	UDP
Handshake	Connection-oriented protocol, where a connection must be established between two devices before transmission	Connection-less protocol, where no connections need to be established before transmission
Ordering	Data is streamed to the client, meaning packets are guaranteed to arrive in the right order	Packets are sent independently from each other, which means they can arrive in any order
Reliability	Packets are guaranteed to be delivered at the destination, being re-transmitted in case of failure	There is no guaranteed delivery of data, leading to the loss of some packets
Error checking	Checks packets for integrity, and recovers erroneous packets	Checks packets for integrity, but discards them in case of corruption
Speed	Slower transfer speed due to intensive error checks and handling	Faster transfer speed as re-transmission and packet recuperation is not done

Table 5.1: Comparison between TCP and UDP.

[7] VMware Workstation, VMware, Inc. (Last accessed: 2020-06-30): https://www.vmware.com/

By analysing the comparison between both protocols, the choice that first comes to mind is UDP, since it is a protocol that is commonly used in image and video streaming services, where the loss of one frame is not problematic because it is going to be replaced by the next frame as soon as it arrives, which is essentially what is trying to be accomplished here. However, the bitmaps that are transferred from the old GUI to the AUI have an approximate size of 3 MB. This means that each bitmap would have to be split in large packets that surpass the size of the Maximum Transmission Unit (MTU), causing the packets to be fragmented. The loss of a single fragment leads to the loss of the whole packet, which is more prone to happen when the UDP packets are large. Furthermore, UDP does not guarantee the arrival of all packets in the correct ordering, as TCP does. All it takes for one frame to become corrupt is for one packet to arrive sooner than it should or to be lost in transit. Through testing with both protocols, it was found that this happened very often when using UDP, which led to a poor visual experience, even though the speed of transfer was higher. Nevertheless, the transfer speed difference between both protocols was not very significant (average of 18.4 frames per second for TCP versus 22.3 frames per second for UDP), which made TCP the choice of communication protocol to be implemented in this solution.

Since TCP requires setting up a connection between the two devices, a handshaking procedure was executed between the AUI and the process responsible for taking screenshots of the old GUI, leading to a *socket* based *client-server* connection, where the old GUI acts as a *server* that sends the interface image data to the AUI, which acts as a *client*. The development library used to implement this connection was Boost C++, as referred in section 5.1, specifically the *ASIO* module of the library. As the communication was established between a physical machine and a virtual machine, a *host-only network* was set up between both machines using the *VMware Virtual Network Editor*, which assigned an IP address to each of the machines. The host-only network has the advantage of only allowing a connection between the physical machine and the virtual machine (which is ideal here since no other connection is required) and it ensures faster transfer of data through the network, since host-only networks use the internal CPU resources to handle the connection.

5.3.1 Server (GUI)

As is usual in a TCP based connection, the server first creates a socket that is bound to a specific IP address. In this case, the IP address bound to the server socket was the one assigned by the VMware Virtual Network Editor to the virtual machine (old GUI). The socket then listens for a connection at a chosen port, waiting until a connection request arrives. When it does, the server accepts the connection and the transmission of data begins.

Before starting a continuous transfer of screenshots from the old GUI, the server sends the size of the image and the bitmap header information to the client. This information is sent only once before

the screenshot transmission begins, since this information is equal for all screenshots. After the client is notified about the size and header info, the cycle of transmission begins. In each cycle, the server uses the *takeScreenshot* method of the *InterfaceScreen* class object associated with the old GUI to save the screenshot information in a byte array, which is then sent through the socket to the client. When the interface is closed, the connection between the two devices is shutdown and the socket is closed. Also, the *InterfaceScreen* class object calls its *destructor*, that proceeds to release the device context memory, as well as the byte array previously allocated to hold the screenshot data.

5.3.2 Client (AUI)

The client behaves similarly to the server, but its actions complement the actions of the server. First, a socket is created and is bound to an IP address in a similar fashion to the server. The client then proceeds to attempt a connection with the server's IP address at the chosen port. After the server accepts the connection, the client waits to receive the size of the screenshots and the bitmap header information, after which the cycle of transmission begins. In each cycle, the client waits to receive the screenshot data array, which is then stored in memory and loaded to the graphical interface that proceeds to render the new interface based on the emotional state estimation (a detailed explanation of the loading and rendering process will be given in section 5.4). After the interface is closed, the connection is shutdown and the socket is closed. As happened in the server, the memory allocated to hold the received screenshot data is also freed.

5.3.3 Error Handling

To guarantee a continuous and reliable transfer of data between the old GUI and the AUI, custom exceptions were created in order to contemplate all possible failures of the software and handle them correctly. Therefore, two types of exception were created: *lightException* and *heavyException*. A heavy exception is thrown when an error code has been generated that compromises the whole execution of the software. In this case, a heavy exception is only thrown when the client-server connection is aborted unintentionally. In this case, the program is forced to shutdown completely since the interface cannot work without the screenshot streaming. On the other hand, a light exception comprehends any error that might occur in the system that only affects the current cycle of transmission. In this work, a light exception is thrown in two situations: when an error code is generated while reading or sending a message the socket that is not indicative of a connection abortion; when there is an error loading screenshots to the SDL interface. When this happens, the current cycle is terminated and the execution continues to the next cycle, thus contributing to the robustness of the system. Apart from these two custom exceptions, the program also catches any other standard exceptions thrown by the standard

libraries, in which case a shutdown is forced, since the cause of the exception is unknown and can generate unwanted behaviour.

5.4 Interface Changes

As the AUI receives the screenshots of the older interface and acknowledges the user's emotional state from the neural network classifier, it then proceeds to render the new interface. Before the rendering process can take action, an SDL initialization procedure must be undertaken. This procedure takes place after the client-server connection is established and size of the screenshots has been received. SDL then creates a window with the dimensions of the original interface and creates a 2D rendering context that allows the rendering of textures to the new interface's window. In this work, a class called *LTexture* was created in order to handle all rendering-related operations. This class contains a data structure of type *SDL_Texture*, which is an SDL structure type that contains an efficient representation of pixel data. The class also features several methods for texture manipulation, allowing the loading of a byte array to the SDL_Texture, the rendering of the whole texture to the SDL window, or even the rendering of specific parts of the texture to an area of the window of choice (a technique known as *clip rendering*), along with resizing and alpha blending, which allows to add transparency to a texture. This plethora of methods permits the redefinition of the original interface in any desired way, allowing the system to perform changes to the interface with respect to the operator's emotional state, thus improving the user's performance in robot teleoperation tasks.

Therefore, after receiving the screenshot from the server, the AUI loads the byte array containing the screenshot data into an LTexture object. Subsequently, the rendering process takes action using the LTexture class features. Depending on the predicted emotional state, the AUI performs the actions described previously in section 3.4.

5.5 Request Processing

In order to enable the AUI to communicate requests to the old GUI, a task automation tool called *SikuliX* was used. SikuliX can find any GUI elements on the screen and perform operations such as clicking, text writing, dragging, amongst others. This tool is very useful in this solution, as it allows the AUI to send requests for a certain task to be performed in the old GUI, such as changing the maximum speed of the robot, as referred in section 3.4.2, thus making it possible for both interfaces to communicate without access to the original interface's source code. SikuliX offers two distinct ways of performing task automation:

1. **Scripting:** Scripts that dictate the workflow of a certain task using the mouse and keyboard can

be created in the *SikuliX IDE*, which provides an easy interface for script creation. These scripts can then be executed directly from the SikuliX IDE or by calling them in the command line;

2. **Programming:** SikuliX can be directly integrated in other software in the form of a feature library that can be used with Java based languages.

Since this work is developed in C++, integrating SikuliX as an external library was not possible, since it requires the usage of Java or a Java based language. As such, the scripting procedure was adopted. However, in this work, it would not be viable to execute the scripts via the SikuliX IDE, since the scripts must be called by the AUI when a change of emotional state is detected, which leaves calling the scripts through the command line as the only option. This can be performed by the AUI through the usage of a *system call*, which passes any command of choice to the command line interpreter to be executed. However, SikuliX scripts have a long start-up time (approximately 8 seconds), due to the necessary loading and pre-interpretation of numerous Java classes that are necessary to running the scripts. This adds a significant amount of latency to the AUI requests, which is undesirable since they should be executed as soon as possible from the moment the emotional state changes. Therefore, the solution found to this problem was to use a new SikuliX feature that is still under development up to the current date, called the *RunServer*. Although there was some reluctance to use it since it is still under development, which means bugs and other anomalies could exist, all the tests performed with this feature revealed no problems at all during execution, which led to it being used in the final solution.

5.5.1 *RunServer* Deployment

With the implementation of the RunServer, the execution of SikuliX scripts is much more efficient, since the responsibility of running the scripts is handled to a server. As such, there is only an initial startup time of about 10 seconds during the launch of the AUI, with the advantage of the scripts being executed almost instantly upon being called, since the initialization overhead is all undergone at launch time. This feature is intentionally developed to run prepared scripts on environments that cannot easily use SikuliX API, such as a C++ programming environment as the one used in this work. The RunServer also allows the scripts to be called from another machine, which is ideal since the scripts are used in the virtual machine (GUI) but are called in the physical machine (AUI).

The requests to the RunServer are made by loading a specific Uniform Resource Locator (URL) in a browser, which is then processed by the RunServer that proceeds to run a particular SikuliX script based on the URL received. As the utilization of a browser is not a viable choice for the same reason the SikuliX IDE was not directly used (the requests need to come from the AUI and not from direct input of the user), a *batch* file that opens an URL without accessing a browser directly was used. This specific batch file is called *callurl*, and it enables the start up, request processing and termination of the

RunServer to be feasible using only commands on the command line, and by extent, on the AUI C++ environment. Below is a list of all the steps performed to enable the processing of requests by the AUI (note that steps 1 and 2 are performed on the old GUI side, while steps 3 to 6 are executed on the AUI side of the application):

1. A folder to hold the SikuliX scripts called *"sikulixrunserver"* is created, which should be located in the "home" folder of the respective operating system (in this case, for Windows XP, the "home" folder is *"<root>/Documents and Settings/<username>"*);
2. The RunServer is started on port 50001 through the usage of the command *"<path-to-sikuli-folder>/runsikulix -s"*. The "*-s*" option informs SikuliX that it should be launched on server mode;
3. The server is notified of the location of the SikuliX scripts by calling the command *"callurl http://192.168.40.150:50001/scripts/home/sikulixrunserver"*. Note that the IP address used corresponds to the IP assigned to the virtual machine, where the old GUI is running;
4. A script runner is started by executing the command *"callurl http://192.168.40.150:50001/startp"*, that will later enable scripts to be started without a startup delay;
5. Scripts are ran with the command *"callurl http://192.168.40.150:50001/run/<script-name>"*
6. The RunServer is terminated using the command *"callurl http://192.168.40.150:50001/stop"*

This procedure enables the AUI to request the execution of specific prepared scripts by the old GUI, which were written using the SikuliX IDE.

5.5.2 *SikuliX* Scripts

For this solution, only two SikuliX scripts were needed: one for increasing the maximum speed of the robot when the neural network classifies the operator's state as stress; another for reverting this change when the classifier predicts the state as rest or workload. In order to change the maximum speed of the robot, the Setup tab of the interface must be accessed. This tab enables the adjustment of multiple features offered by the robot, such as arm velocity, communication setup, docking setup, and also the control of the maximum velocity of the robot. The interface includes a slider that allows the regulation of this parameter, as can be seen in figure 5.1, which displays the Setup tab of the RAPOSA's original user interface.

The slider highlighted by the yellow rectangle controls the maximum velocity of the robot, which can range from 20 cm/s to 36 cm/s. In order to change this parameter, SikuliX scripts must first access the Setup tab of the interface, then proceed to change the maximum velocity by clicking in a specific point on the slider, and finally return to the Operation tab, so the operator can continue to perform the teleoperation task. Both the script for increasing the maximum velocity and the script to revert that

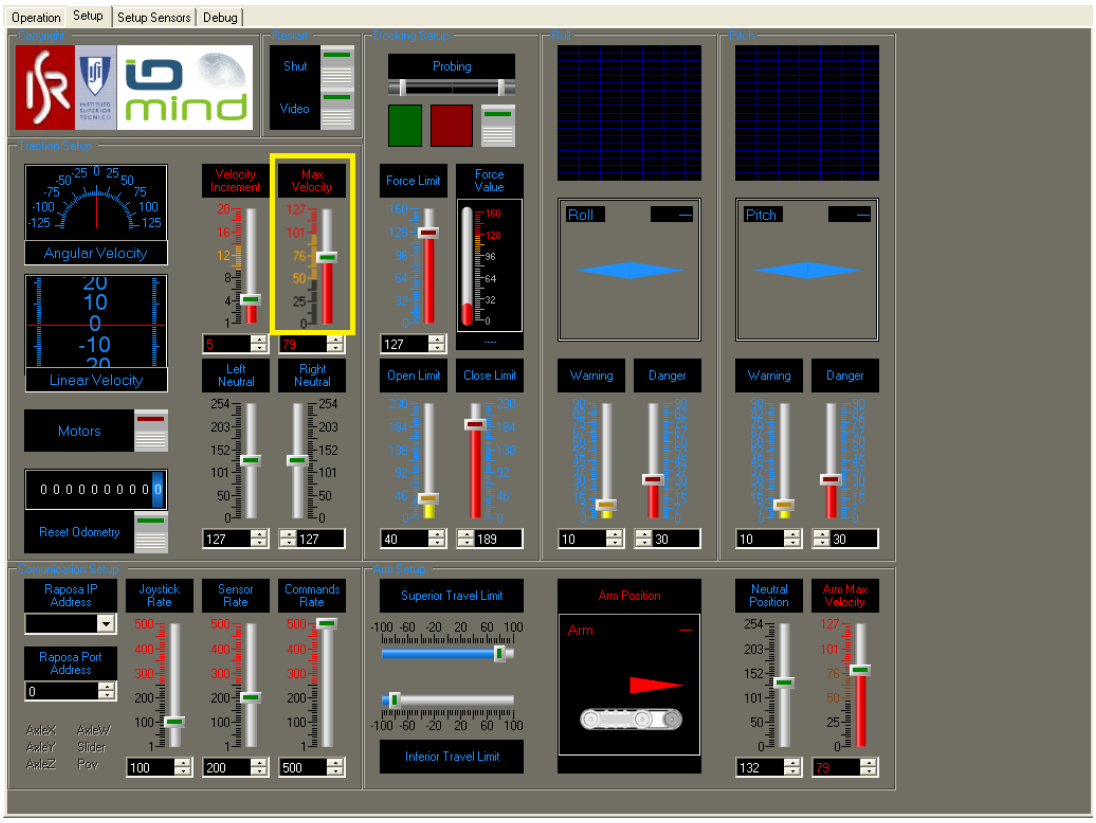


Figure 5.1: Setup tab of the RAPOSA's interface.

change are very similar, the only difference being the value that each one sets on the maximum velocity slider. Below is the list of steps executed by the scripts when called by the AUI:

1. Set the mouse delay to zero (this allows the mouse clicks automated by SikuliX to be executed as fast as possible);
2. Click on the Setup tab at the top of the interface
3. Click in the desired place in the slider in order to change the maximum velocity (when stress is detected, this value is set to 34.2 cm/s, while in rest or workload the default value of 29.5 cm/s is set)
4. Click on the Operation tab at the top of the interface

This procedure enables the AUI to send a request to the old GUI when an emotional state change is detected (except when it changes between rest and workload, since here the maximum velocity is the same), causing the respective SikuliX script to be executed, which leads to a change in the maximum velocity with only three automated clicks.

5.5.3 Performance Analysis

For the purpose of increasing the execution speed of the SikuliX scripts, they were programmed to click directly in specific coordinates of the screen instead of using image recognition to find the areas to be clicked. This approach assumes that the old GUI window will always be on the same position on the screen, which is guaranteed by using the Win32 API function *SetWindowPos*, that sets a specific window position on the desktop screen. This solution improves the execution speed of the scripts, taking between 140 ms to 220 ms to perform each click, yielding an average of 540 ms to execute each script, which is much faster than the average 3 seconds taken when using image recognition. Furthermore, there is a window of time where the operator cannot see the Operation tab, which accounts for the time taken since the Setup tab is selected to the moment the maximum velocity is changed plus the time taken until the Operation tab is selected again. This time window dropped from approximately 2 seconds to 360 ms when using the coordinates of the screen to indicate the areas to be clicked, instead of the image recognition tools provided by SikuliX, which means it is almost unnoticeable during the robot teleoperation task.

6

Evaluation

Contents

This chapter presents the procedures undergone to evaluate the solution proposed in this dissertation, which included the execution of a user study that enabled the testing of the proposed solution's performance, along with the integration of the user's opinions and preferences on the results. First, a brief contextualization of the study performed is given, allowing an understanding of some of the decisions that had to be made regarding the study conditions. Afterwards, a description of the full setup and apparatus used for the study is provided, followed by an explanation of the robot teleoperation tasks performed by the users in order to evaluate the system. Further on, the metrics used for the evaluation of the system are described, along with the procedure adopted and the information relative to the subjects that took part on the experimental sessions. Finally, an account on the results obtained in the study is presented, followed by their analysis and respective discussion.

6.1 Experiment Contextualization

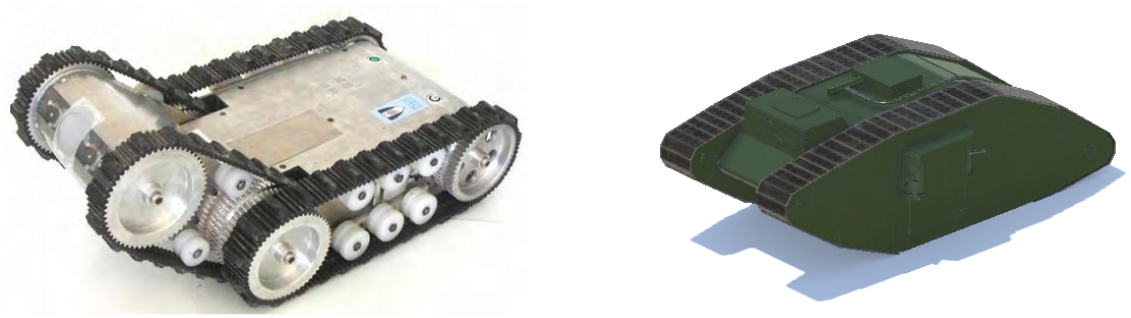
In order to evaluate the proposed solution, the dataset acquired by Singh et al. (2018) [6] was initially used to train a neural network model that would later classify the emotional state of the users. This dataset contained biometric data from five different subjects, that was acquired while subjects were performing tasks designed to induce each of the three emotional states (rest, stress and workload). However, as physiological data varies significantly from person to person (see section 6.7.1), this meant the system could only be evaluated on the same subjects from whom data was collected, which unfortunately were not available to evaluate the present solution. Therefore, a data collection session was carried out in order to acquire a new dataset from subjects that could later participate on the evaluation of the system, meaning each subject participated in two experimental sessions (a first one to acquire a new dataset and a second one with the purpose of evaluating the system with emotional state classification in real-time). Moreover, the high variability inherent to physiological data would require an extensive work to overcome the difficulty of training a classifier using data from all subjects. Even though the usage of person-specific classifiers requires the training of a model for each operator, it was the approach adopted since the emotional state classification is not the main focus of this work.

Additionally, due to the recent global circumstances caused by the COVID-19 outbreak, safety measures had to be prepared in order to mitigate the risk of contagion of the virus over the course of the experimental sessions. A description of the efforts made in this regard are presented in appendix B.

On a separate note, the experimental sessions designed to evaluate the system required subjects to pilot RAPOSA through different environments (a detailed description of the experimental procedures is given in section 6.5). Unfortunately, RAPOSA was not available for usage in this work due to the need of heavy maintenance, which led to the development of a USAR environment simulator in order to enable the evaluation of the proposed solution.

6.1.1 USAR Simulator

The USAR simulator was developed using Unity[1] (version 2018.4.23f1), a cross-platform game engine developed by Unity Technologies. The aim of the simulator was to create a *mock-up* of the RAPOSA's original interface, as well as to develop custom Search and Rescue scenarios that could be used to induce stress and workload to users while performing a robot teleoperation task. In order to do this, a simulated tracked robot developed by Corujeira[2] (2018) called *MarkV* (inspired by the MarkV tank) was imported to Unity. The robot's controllability and appearance are similar to the RAPOSA's, although it does not feature an arm control (see figure 6.1). Nevertheless, MarkV can still climb stairs and obstacles like RAPOSA would do, only without having to manually control the arm.



(a) RAPOSA robot.

(b) MarkV simulated robot.

Figure 6.1: Comparison between RAPOSA and the simulated robot used for evaluation purposes.

After importing the MarkV robot to Unity, the mock-up of the RAPOSA's interface was created using the Unity *Canvas* game object. Although the original interface's appearance was maintained, only some of its functionalities were incorporated. Below is a list of the robot's features that were integrated in the simulator:

- Pitch and roll indicators, which are regulated by the rotation values of the MarkV *game object*;
- Lights intensity, which is controlled by a *spotlight* object that is attached to the robot's front;
- Motor and electronics batteries, where the motor battery gets depleted according to the movement of the robot and the electronics battery has a constant rate of decay that is increased linearly in function of the robot's lights intensity;
- Linear and angular velocity indicators, whose values are obtained directly from the MarkV game object;

[1] Unity Real-Time Development Platform, Unity Technologies (Last accessed: 2020-07-02): https://unity3d.com/pt/get-unity/download/archive

[2] Unity Robot Simulation, J. Corujeira, 2018 (Last accessed: 2020-07-02): http://dante.isr.tecnico.ulisboa.pt/jcorujeira/unity-robot-simulation

- Temperature sensor, where the temperature rises over time according to the distance between the robot and nearby fires;
- Front and rear camera, which are obtained using a *camera* object at the robot's front and rear;
- Maximum velocity regulator in the Setup tab of the interface

Overall, the main functionalities of the original interface were maintained, with the exception of the thermal camera, the arm control and the sensor readings (where only temperature was included).

Apart from re-creating the interface in a manner that can simulate the operation of RAPOSA as close as possible to reality, Search and Rescue environments were also created in order to evaluate the system. These environments were developed with the aid of the *ProBuilder*[3] and *ProGrids*[4] Unity packages, which enable the modelling and texturing of 3D geometry. In addition, some free assets were imported from the Unity Asset Store, such as furniture models from the Furnished Cabin pack[5] and multiple textures and materials from the Yughues packs[6]. These tools allowed the creation of a home-like simulated environment, which was used in the planning of robot teleoperation tasks designed to induce different emotional states in the subjects (see section 6.3).

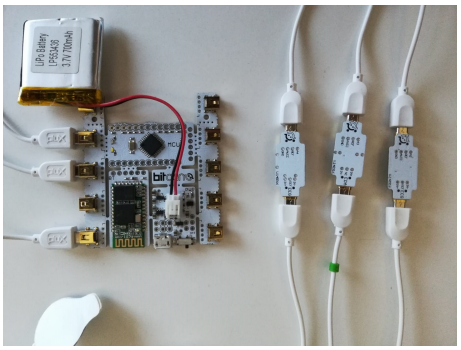
6.1.2 Subject Grouping

For the purposes of the second experimental session (evaluation session), the subjects were split in two independent groups, where one group tested the PAUI approach and the other group tested the classic GUI approach. This method leads to higher variance in the results obtained, since the task performance can be dependent on each user's natural tendency to control the robot more easily than others. However, having each subject test both approachs would not be feasible, since the usage of the same environment for the evaluation of both approaches would lead it to be influenced by carryover effects. This would cause users to naturally perform better in the second approach, whichever it was. A way to work around this problem would be to use two different environments for each approach, but it would yield biased results, since one environment could possibly be easier to operate in than the other. As such, the adopted solution was to follow a between-groups design, where each group tests a different approach. In order to mitigate the variance in the results as much as possible, the Immersive Tendencies Questionnaire (ITQ) [41] was used, which reliably reflects each subject's tendency to become more involved in virtual environment tasks. As subjects with higher scores can potentially have a better performance, the scores obtained in the questionnaire were then used to split the subjects in two balanced groups, where each group tested a different approach in the second experimental session.

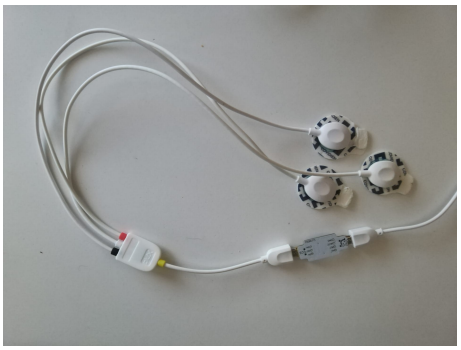
[3] ProBuilder Package (Last accessed: 2020-07-03): https://docs.unity3d.com/Packages/com.unity.probuilder@4.2/manual/index.html
[4] ProGrids Package (Last accessed: 2020-07-03): https://docs.unity3d.com/Packages/com.unity.progrids@3.0/manual/index.html
[5] Furnished Cabin Pack (Last accessed: 2020-07-04): https://assetstore.unity.com/packages/3d/environments/urban/furnished-cabin-71426
[6] Nobiax / Yughues Free Materials and Textures (Last accessed: 2020-07-03): https://assetstore.unity.com/publishers/4986

6.2 Apparatus Used and Setup

As pointed out previously in section 3.1, the devices used to extract data from users were the Bitalino (r)evolution Plugged Kit BT for physiological data (figures 6.2(a) and 6.2(b)), the Tobii 4C Eye Tracker for eye movements (figure 6.2(c)) and the laptop integrated webcam for facial expressions and emotions.



(a) Bitalino board with ECG, EEG and EDA sensors plugged in.



(b) Set of electrodes attached to a 3-lead accessory.

(c) Tobii 4C Eye Tracker.

Figure 6.2: Devices used for extracting physiological signals and eye movements.

In order to collect physiological data, the ECG, EEG and EDA electrodes had to be placed in the subjects in specific configurations. For ECG, the best suggested placement by Němcová et al. (2016) [42] was used: positive lead under right clavicula, negative lead under left musculus pectoralis major and reference lead under left clavicula. For EEG, the positive and negative leads were placed at forehead and the reference lead at the left earlobe. For EDA, only two electrodes are required, which were placed in the left hand palm [6].

Apart from the data acquisition devices, the Krom Key gamepad[7] (figure 6.3) was used to control the robot in the simulated environment, since RAPOSA is also controlled by a gamepad in reality. The gamepad provides a joystick control that was used move the robot, as well as the top right and left buttons that were used to adjust the robot's light intensity.

[7]Krom Key Gamepad, Krom Gaming (Last accessed: 2020-07-05): https://www.kromgaming.com/en/controllers/key

Figure 6.3: Krom Key gamepad.

Figure 6.4 displays the complete setup used for the experimental sessions designed to evaluate the solution presented in this dissertation:

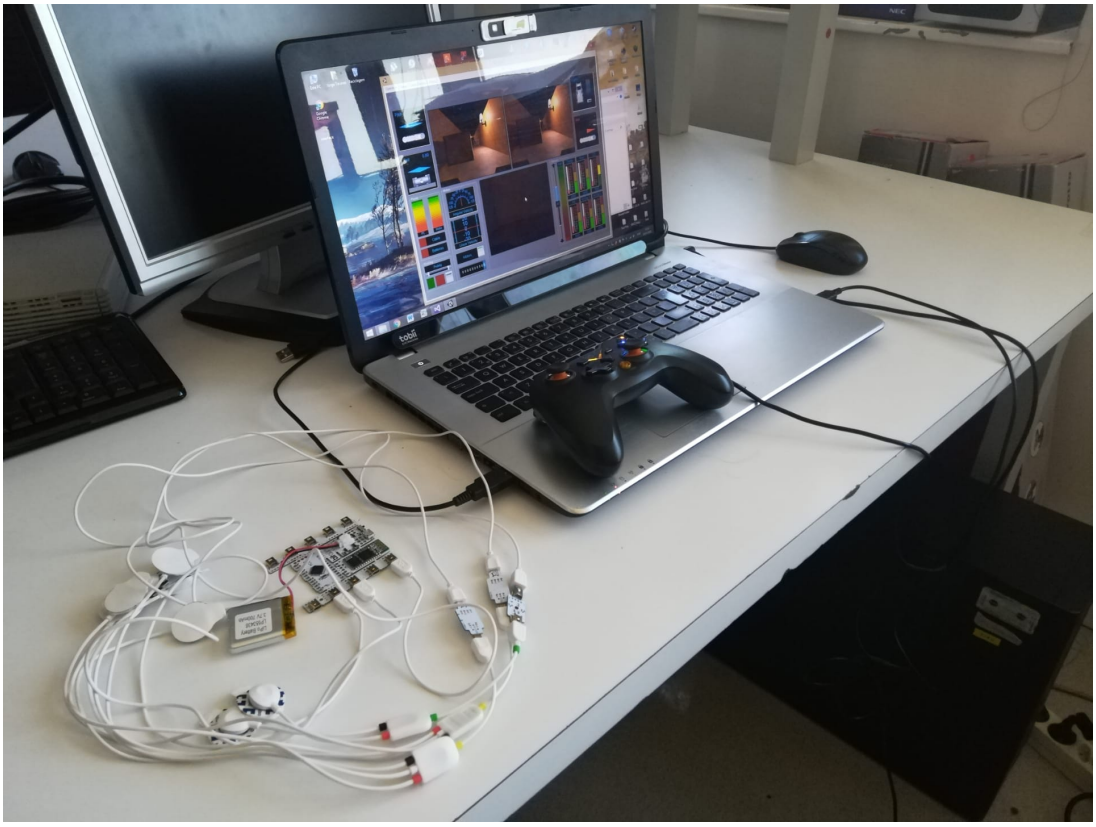


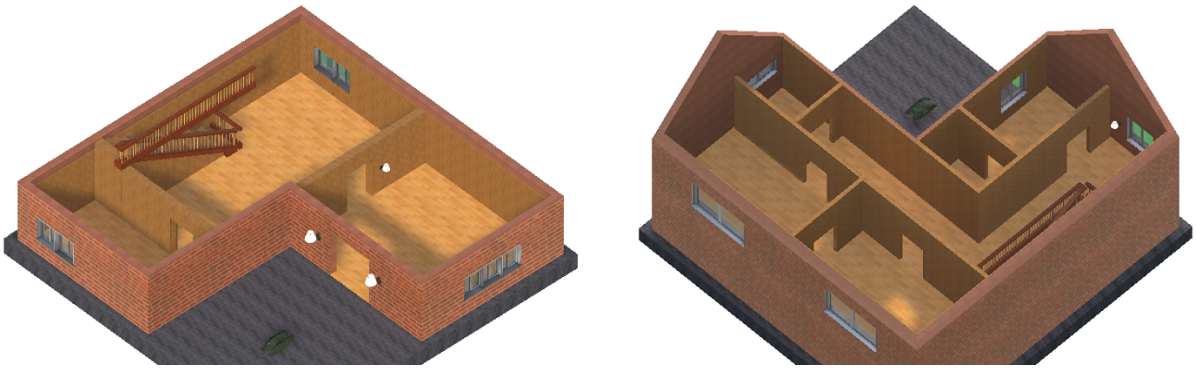
Figure 6.4: Full experimental setup, including the robot's interface, along with the gamepad, the eye tracker attached to the lower part of the screen and the Bitalino board on the side with the three sensors and respective electrodes.

6.3 Performed Tasks

In order to collect data to train a classifier and later use it to predict the emotional state of a person among rest, stress and workload, a task meant to induce each of these states was designed using the developed USAR simulator. All three tasks required the subject to drive the robot in a home-like setup while attempting to complete a certain objective, where the home is adapted to fulfill the needs of each task described below. In addition, a training task was also planned with the purpose of familiarizing the subjects with the robot teleoperation process.

6.3.1 Rest Task

During the rest task, each participant was required to drive along a room inside an empty house environment for five minutes. During this task, the light conditions were favorable and the batteries did not discharge, meaning the participant did not have to worry about battery or light management. By attributing an easy objective and relieving the participant of all possible worries on the interface, the task does not require great mental or physical effort, thus leaving the participant in a restful state, possibly with windows of boredom. Figure 6.5 represents an aerial view of the first and seconds floors of the home environment during the rest task:



(a) First floor (b) Second floor

Figure 6.5: Aerial view of the simulated home environment during the rest task

6.3.2 Stress Task

In the stress task, participants were asked to find four victims inside a house on fire before a timer ran out. The same home used in the rest task was used here, but instead of being empty, some features were added in order to increase the difficulty of teleoperating the robot. Below is a list of the stress-inducing events that occur during this task:

- A specific timer to find each victim is set. When the timer reaches zero, the nearest victim to the robot at that moment becomes trapped, becoming impossible to save. Whether the timer reaches zero or the victim is found, a new timer is set for the next victim, that is increasingly smaller (4.5 minutes for the first victim, 4 minutes for the second, 3 minutes for the third and only 1 minute for the last victim). The timer values were determined through pilot testing;

- The timer is displayed in the interface and a beep is played as each second passes, in order to remind the participant that the time is running out. In the last thirty seconds of each timer, the beep frequency is doubled in order to intensify stress;

- The house is fully furnished, but the furniture is out of place and blocks some paths;

- As would happen in a real fire house, piles of wooden logs and rubble are scattered around, essentially constituting obstacles that make operating the robot much more difficult. Participants can often get stuck in these obstacles and the process of getting unstuck can be very stressful;

- Fire spots are located in some places, meaning the participant needs to guarantee the temperature does not get too high (especially stressful when participants get stuck in an obstacle that is located near a fire spot). The temperature sensor allows the participant to visualize the current temperature value, along with the danger limit;

- As the participant goes through specific places, wooden logs can fall off the roof, making a very loud noise and blocking some paths, forcing the participant to choose alternative paths;

- The simulation occurs during the night, meaning the light conditions are precarious, although the fires contribute to some areas being more lit up;

- Flickering light bulbs were added in some corners to cause visual distress;

- A docking post (represented by a green cube) is placed at the house's main door that recharges the robot's batteries if the robot gets near it, thus adding the worry of eventually getting back to recharge batteries. If the participant lets the batteries run out, the robot is placed back at the house entry and a 1 minute penalty is applied;

An aerial view of the house during this task is presented in figure 6.6:

(a) First floor (b) Second floor

Figure 6.6: Aerial view of the simulated home environment during the stress task

It should be noted that the light conditions in figure 6.6 have been enhanced in order to facilitate the view of the house configuration. Figure 6.7 shows the timer displayed in the interface during this task, as well as a victim counter that lets the participant know how many victims have been found so far:

Figure 6.7: View of the interface displayed during the stress task, showing a timer in the top and a victim counter in the bottom

6.3.3 Workload Task

In the workload task, participants were required to find 10 objects (represented by red cubes) inside a house while answering workload inducing questions. In this task, the house was furnished but not on fire, making the environment easier to traverse in comparison to the stress task. However, the furniture was kept as a way to hide the objects inside the house (objects were often placed behind closets or fallen tables). Additionally, the house is very dark and the robot's spotlight intensity was limited, in order to prevent participants from seeing objects from very far, thus increasing the work required to find them. In order to induce cognitive workload on the participants, a set of questions was prepared, which include:

- Basic arithmetic operations
- Requests to read values of sensors in the interface
- Questions about the surroundings of the robot
- Finding the missing number in a sequence of numbers
- Solving basic logic problems with images

During this task, the logic problems and the sequences of numbers were displayed side to side with the interface, in order to facilitate their visualization by the participant. The list of problems used can be found in appendix A, along with the set of questions that were asked over the course of the task. The participants were told to refer to each specific sequence or problem when asked one of these types of questions. It should also be noted that participants were reminded that there was not a time limit to answer each question, in order to avoid inducing unnecessary stress. Also, similarly to the victim counter presented in the stress task, the interface displayed a counter for the number of objects found.

Figure 6.8 below shows an aerial view of the environment during this task:

(a) First floor (b) Second floor

Figure 6.8: Aerial view of the simulated home environment during the workload task

6.3.4 Training Task

During the training task, participants were asked to drive the robot through a path filled with increasingly harder obstacles, requiring the subject to reach the end of the path in order to complete the task. Along the path, apart from finding obstacles similar to the ones encountered in the tasks described previously, the subject was also required to understand all the interface elements. The path forced the user to drive close to fires, where he/she would be warned about the need to pay attention to the robot's temperature. There was also a very dark zone along the path, where the user had to regulate the robot's light intensity in order to cross it successfully, finding a docking post at the end that showed how the batteries can be recharged. At the end of the path, a message was prompted in the screen informing the user that the training task was over, although he/she was encouraged to stay in the training environment for longer until feeling confortable with the teleoperation of the robot.

6.4 Metrics

In order to evaluate the hypotheses formulated in section 1.2, two types of metrics were considered: task performance metrics and user experience metrics. Task performance refers to the quality of the tasks achieved by users, where speed and accuracy are typically the most important metrics. Additionally, it was also of interest to measure the level of attention that users retained when performing the tasks. Therefore, the task performance metrics defined were:

- **Completion time:** The time measured since the beginning of the stress task until its end, for each subject. The stress task was chosen for this measurement since it is the only task where time plays an important role in dictating the performance of the user;
- **Number of objects found:** The number of objects found by each subject during the workload task. Over the course of the evaluation session, the user is not informed that there is a time limit on the task, making the number of objects found a measure of the effectiveness of the subjects;
- **Engagement levels:** The relative change of the mean engagement values from the rest task to the workload task, in percentage. According to McMahan et al. (2015) [43], the engagement index that is extracted from the EEG sensor (as seen in table 3.1) reflects a person's ability to sustain attention and gather information. Here, instead of using the absolute value as a metric, which shows differences in its base value depending on the subject, the relative change observed in its value in relation to the baseline was used.

Regarding user experience metrics, they usually refer to the subjective perception of the system reported by each user, which is usually done through questionnaires or interviews, and can either be quantitative or qualitative. In this case, the USE Questionnaire [44] was used to measure the level of

usefulness and satisfaction reported by the subjects towards each approach. As the author suggests, a short version of the questionnaire was built by picking some of the most heavily weighted questions that fit the context of the system to evaluate.

It should be noted that, even though it does not contribute as a metric, the Discrete Emotions Questionnaire (DEQ) [45] was also filled by the participants for purposes of discussion, which was aimed at understanding what type of emotions were felt by the user over the course of each task. The DEQ is suitable for measuring self-reported emotions that are associated with specific emotional states, which the author refers to as *subscales*. Following up on the author's recommendation, only the *anger*, *anxiety* and *relaxation* subscales were included in the questionnaire, since they are the ones that most accurately represent the emotional states that subjects were expected to feel during the robot teleoperation tasks. Additionally, the subjects that tested the PAUI approach answered additional questions regarding the attentive aspect of the interface, focusing on the characterization of the changes performed by the interface and their preferences towards the system's capabilities.

6.5 Procedure

For the purposes of the system evaluation, a preliminary experimental session was undertaken to collect physiological data from the subjects, with the aim of assembling a new dataset. The collected data was used to train an emotional state classifier that was employed in the evaluation of the approach with the same subjects. In the evaluation session, one group of subjects tested the classical GUI approach and the other group tested the PAUI approach, enabling a comparison of the performance obtained by both groups in the robot teleoperation tasks.

Both sessions had a very similar procedure, which took aproximatelly 1 hour to complete. Upon arrival to the testing office, the participant sat in a chair in front of the testing setup and was given a description of the experimental procedure. In order to guarantee that all subjects received the same information regarding the context of the study, a small text describing the experiment was written previously, serving as a guideline for the supervisor. In this phase, the subject was given a contextualization of the problem and a description of the three types of robot teleoperation tasks he/she had to perform while biometric data was being acquired. An overview of the robot's GUI, its controls and its capabilities were then given. Finally, the participant was also reminded to put his/her phone in airplane mode in order to avoid distractions during the experimental session.

After the introductory briefing, the subject was asked to sign a data collection consent form, authorizing the extraction of biometric and image data for research purposes, which also ensured his/her anonymity. Subsequently, the participant proceeded to fill a demographics questionnaire, followed by the Immersive Tendencies Questionnaire (this step was exclusive to the first experimental session). In

order to maintain anonimity, a number ID was attributed to each subject that was used to identify each person's answers and the respective collected data.

Afterwards, the eye tracking device was calibrated for the subject using the profile creation tool in the Tobii Eye Tracking software. Before the calibration, the subject was asked to pick a comfortable position and to try not to deviate away from it too much throughout the experiment, since the calibration of the eye tracker relies on the positioning of the user (this was merely a prevention measure, since the eye tracker can still detect eye movements accurately when the user's position has a small offset). Alongside, the laptop was also positioned in a way that guaranteed the subject's face was relatively centered in the images captured by the camera. The light conditions in the room also ensured that the facial expressions were captured effectively by the Affectiva API.

Subsequently, the ECG, EEG and EDA electrodes were attached to the subject according to the configurations referred previously in section 6.2 (see figure 6.9). Then, the signals extracted were tested using the Bitalino data visualization software (*OpenSignals*[8]) in order to ensure the ECG, EEG and EDA waveforms were being traced as supposed. Additionally, the PAUI signal extractor was also tested in order to ensure that the data files were being output correctly.

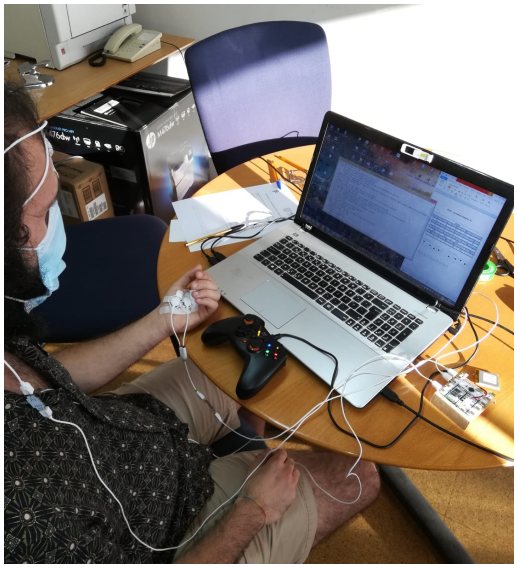


Figure 6.9: Subject with attached electrodes ready to start the robot teleoperation tasks.

After going through all the necessary preparations, both the signal extractor and the simulator were started. Beyond this point, there was no further communication between the supervisor and the subject, with the exception of the workload task that includes a set of workload-inducing questions, as referred in section 6.3.3. From here on, all instructions were given by the simulator itself, that prompted messages indicating what the user should do at any given time.

[8]OpenSignals (r)evolution, Plux (Last accessed: 2020-07-07): https://bitalino.com/en/software

At first, the subject went through a training session in order to get acquainted with the teleoperation of the robot, which usually took 10 to 15 minutes to complete. After leaving the training session, the order of the tasks was defined randomly by the simulator. It should be noted that, due to the stress task taking on average twice as long in comparison to the other tasks, the rest and workload tasks were executed two times each during the data collection session, in order to guarantee that the obtained dataset was class-balanced. In order to prevent the user from recognizing the environment in the second iteration, a variation of the workload task was performed instead, where the home furniture arrangement and the positioning of the objects was changed. As for the rest task, the user was only allowed to drive on the first floor of the house in the first iteration and on the second floor in the second iteration. It should also be pointed that the order defined by simulator guaranteed that the two rest or workload iterations did not happen successively.

After completing each task, the user went through a 2 minute break period timed by the simulator, in order to relax and come back to a normal emotional state. This period was extended whenever the user did not feel ready to start the next task. Before each task started, the simulator would also give a brief description of the task that followed, in order to inform the user of which of the three tasks he/she was going to perform next. After completing the last task, the signal extraction was stopped and the electrodes were detached from the user.

Regarding the second experimental session, after finishing the teleoperation tasks, the subject was also asked to fill the USE Questionnaire and the DEQ, followed by questions relative to the attentive elements of the interface, in case the subject belonged to the PAUI group. It should also be noted that, in case the subject belonged to the group that tested the classical GUI approach, the emotional state classifier still predicted the emotional state of the subject in real-time for the purposes of evaluating the classifier itself, but the system did not act on this information. In addition, the home environment was slightly modified in comparison with the first experimental session, in order to prevent the subjects from remembering the previous environments. For the same reason, the set of workload-inducing questions was remodeled as well.

The procedure can then be summarized as follows:

- **Introduction**:
  - Contextualization of the experiment;
  - Filling of the consent form, the demographics questionnaire and the ITQ (first session)
  - Calibration of the eye tracking device;
  - Attachment of electrodes to subject;
- **Training Session**:
  - Start-up of the simulation and signal extraction

  - Execution of the training task;

- **Robot Teleoperation Tasks**:
  - Definition of the order of tasks;
  - Execution of one of three possible tasks (rest, stress or workload);
  - Break period with a minimum of 2 minutes between each task;

- **Experiment Conclusion**:
  - Stopping of the signal extraction;
  - Dettachment of the electrodes;
  - Filling of the USE Questionnaire, the DEQ and the interface changes questionnaire for the PAUI group (second session);

6.6 Participants

The subject group used in this study included 6 voluntary participants (4 male, 2 female) aged between 20 and 24 years old (M = 22.8, SD = 1.5). All participants were Portuguese university students, with a background in engineering and without prior experience in robot teleoperation.

6.7 Results

This section presents the measures adopted to visualize the data acquired during the first experimental session, as well as the comparison of the classification accuracies obtained from different neural network model iterations. Furthermore, the ITQ results that led to the group division are also presented, as well as the statistical test results obtained from the metrics recorded in the second experimental session.

6.7.1 Model Training

After collecting the extracted parameters from the six subjects that participated in the first experimental session, a visualization of the data was carried out in order to understand better the patterns present in the data. In conformity with the findings of other researchers [22] [23], it was found that there is an inherently high variability in the extracted physiological data, particularly from subject to subject, which arises from the fact that different subjects might be prone to different emotional reactions due to their personalities and experiences. The variance of physiological data between subjects was made evident through the application of a t-SNE to a dataset formed by the acquired data of all six subjects.

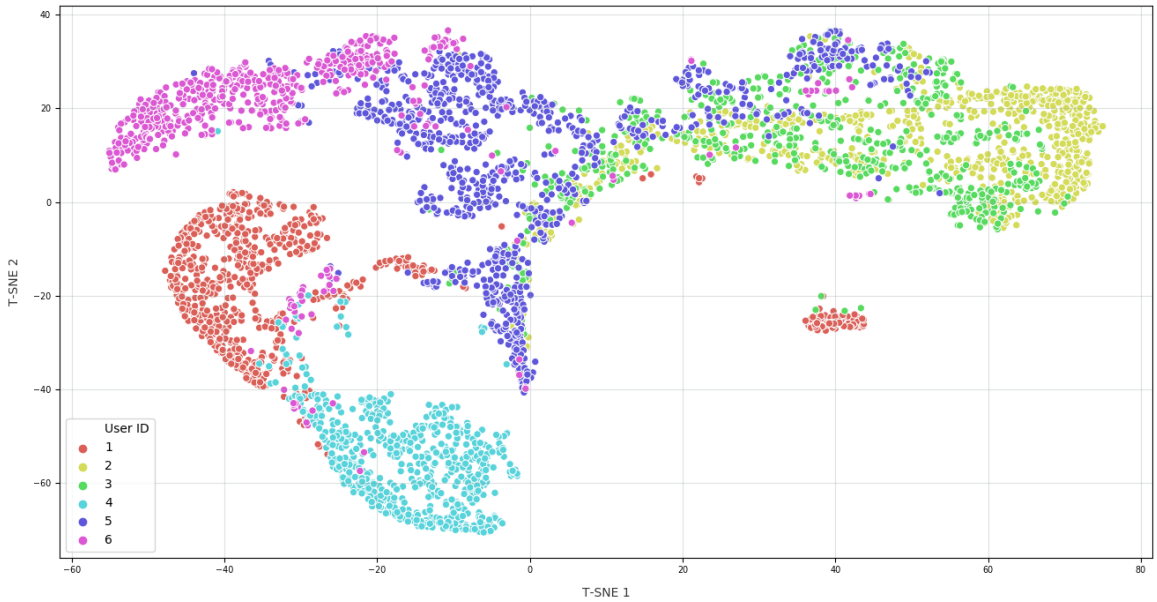


Figure 6.10: Visualization of the two-dimensional embedded space that results from the application of t-SNE (Perplexity = 40, Number of iterations = 2000) to the data collected from each user during the stress task, where each user is represented by a different color.

As can be seen in figure 6.10, the clusters formed by the t-SNE tend to group data from each subject separately from the data of other subjects, showing that there is a significant inter-subject variability present in the dataset. For this reason, as referred in section 6.1, the approach adopted for the classification of the emotional states was to train a neural network for each individual subject, using only the data from that subject, as opposed to training a single model using the data from all subjects. It should also be noted that the features extracted from the Tobii 4C Eye Tracker did not correlate well with the emotional state classification task, since most of them are relative to fixation coordinates and durations. Additionally, even though the Tobii thread operates at 90 Hz, the signal extractor developed by Singh (2018) [6] is only able to write extracted eye tracking features when the Tobii API generates an event related to fixation changes, which happens in intervals of approximately 5 seconds, meaning that most data samples in the dataset had missing eye tracking data, since the sampling period for physiological signals and facial expressions is 1 second. For these reasons, the eye tracking features were not considered in the emotional state classifier training.

Given the high dimension of the feature space (45-dimensional array), a PCA was applied to the dataset, as explained in section 4.7. In order to choose the optimal number of principal components to keep, a *scree plot* (plot of the eigenvalues of the principal components) was charted, showing the proportion of explained variance of each principal component, along with the cumulative proportion of explained variance. As can be seen in figure 6.11, the components are ordered by their relative importance in terms of the variability they explain in the data. Due to the inherent tendency of physiological

signals to contain large amounts of noise [24], it was decided that it would be best to keep most of the explained variance present, in order to retain most of the valuable information. With this purpose, 95% of the total variance in the data was kept, which is explained by the first 31 principal components. The fact that the dimensionality of the dataset was reduced from 45 to 31 while still keeping 95% of the information shows that many features did not contribute actively to the estimation of an emotional state. However, any contribute these features gave to the overall estimation of the emotional state was condensed in the principal components that were kept.

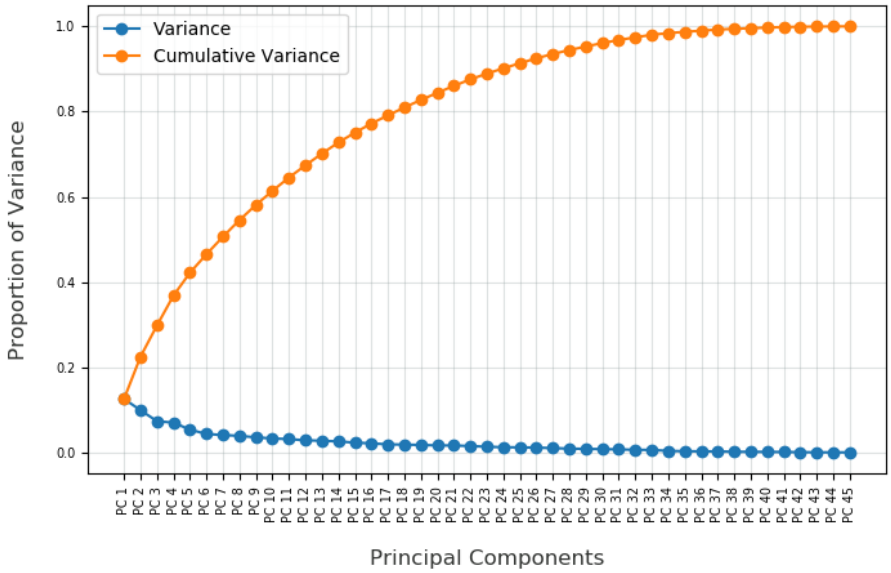
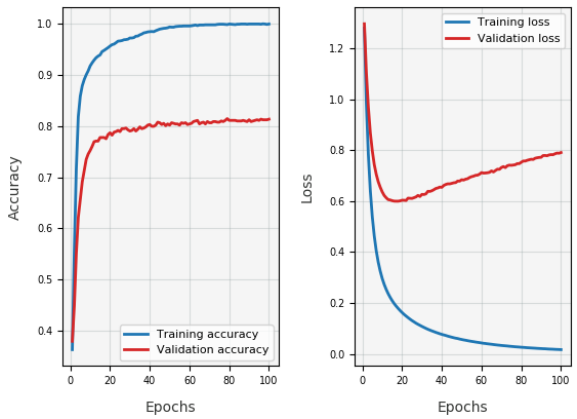
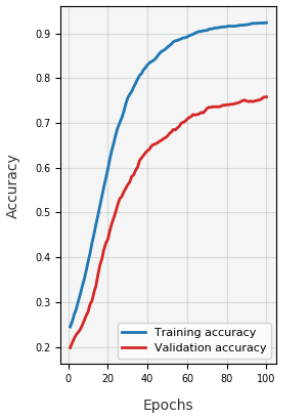


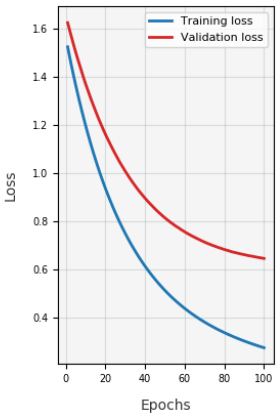
Figure 6.11: Relative explained variance and cumulative variance as a function of the principal components.

After projecting the dataset on the new feature space, the model training procedures discussed in chapter 4 were carried out. As a starting point, a model with a single hidden layer with 40 neurons was built. The choice of 40 neurons arises in order to avoid bottlenecks of information, which could happen if the number of neurons was smaller than the dimension of the input layer, which is 31. Note that all the following procedures regarding the training process of a model refer to the model trained for a single subject (in this case, user 2), since the procedure adopted was similar for all subjects. The activation function chosen for the hidden layers was the *rectified linear unit* (ReLU), which is a popular choice of activation function that usually shows better convergence performance than the sigmoid and hyperbolic tangent activation functions, while also providing a solution to the *vanishing gradient problem* [46]. Additionally, an initial value of the batch size of 32 was chosen, as recommended by Bengio (2012) [47]. The learning rate was the first parameter to be optimized, which was tested on 3 different values: 0.001, 0.0001 and 0.00001. Figure 6.12 shows the loss and accuracy graphs obtained for each of these values.

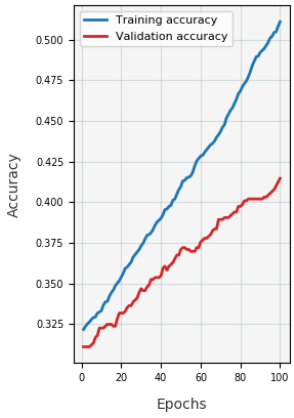


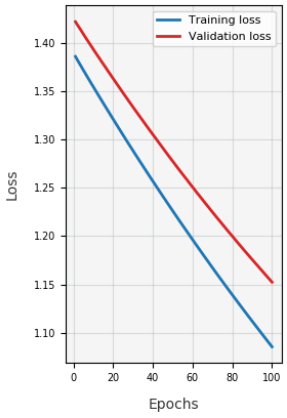
(a) Learning rate of 0.001





(b) Learning rate of 0.0001



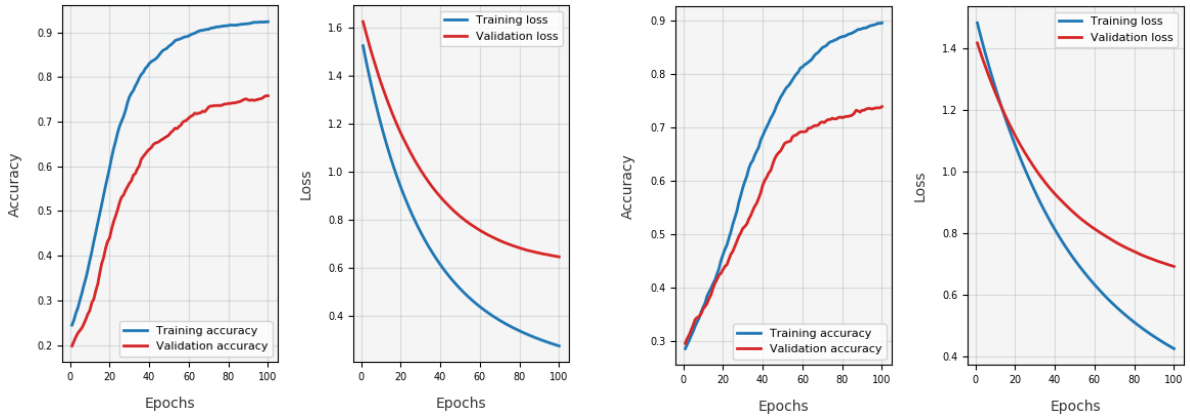


(c) Learning rate of 0.00001

Figure 6.12: Training and validation accuracy and loss for three distinct values of the learning rate hyper-parameter.

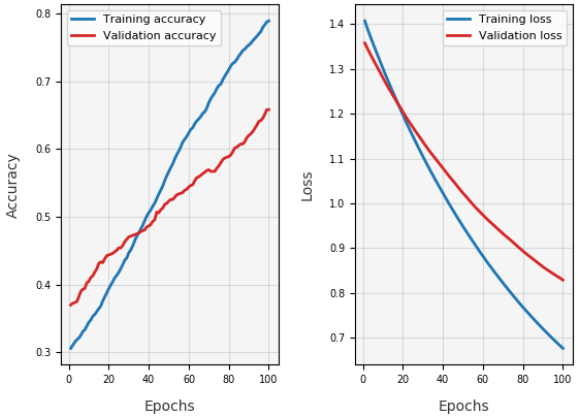
As can be seen, the best value for the learning rate out of the three is 0.0001, where the model achieves an accuracy of 75.8% on the validation set. When the value is 0.001, the model starts overfitting very soon, which is prone to happen with high learning rates, since the gradient descent steps are very large and can lead to exploding gradients. When the value is 0.00001, the speed of convergence is very slow, requiring more computation time to reach the same objective. Subsequently, three values were tested for the batch size: 32, 64 (initial value) and 128. The chosen values are all powers of 2 due to the alignment of computations onto the physical processors of the GPU, which leads to faster computation times. Figure 6.13 shows the loss and accuracy graphs obtained for these values of batch size.

Looking at the graphs, it is clear that the best value for the batch size is the default value of 32. It is also visible that as the batch size increases, the model needs a larger number of epochs of training to achieve lower values of loss, since each epoch performs less gradient steps. While larger batch sizes give a better estimate of the "true" gradient of the loss function, smaller batch sizes can be beneficial in



(a) Batch size of 32

(b) Batch size of 64

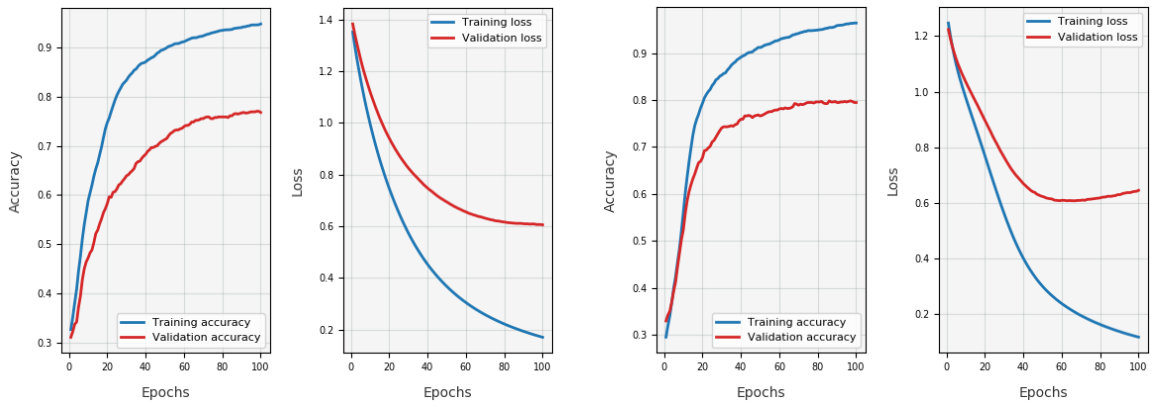


(c) Batch size of 128

Figure 6.13: Training and validation accuracy and loss for three distinct values of the batch size hyper-parameter.

guaranteeing the optimizer will not be as likely to get stuck in local minima, as the gradient estimate can often point in wrong directions, thus leading to better generalization power. After defining a batch size and a learning rate for the optimizer, the network's complexity was increased by adding hidden layers until the overfitting point was reached. After adding a second hidden layer with 30 neurons, the model still did not reach the overfitting point, which led to the addition of a third hidden layer with 20 neurons. The results obtained for these architectures are shown in figure 6.14, where the validation accuracy increased to 77.0% and 79.8% for the architectures with two and three hidden layers, respectively. It can also be seen that the overfitting point is reached around epoch 60 for the architecture with three hidden layers.

After this point, the regularization techniques discussed in section 4.6 were applied separately, namely L^1 regularization, L^2 regularization, and dropout. Early stopping was only applied in the last model, in order to reduce the number of epochs to the point where overfitting starts. Both for L^1 and



(a) Two hidden layers (40 and 30 neurons)

(b) Three hidden layers (40, 30 and 20 neurons)

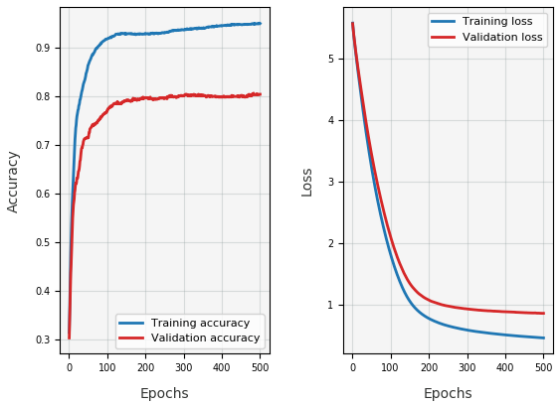
Figure 6.14: Training and validation accuracy and loss after the addition of hidden layers to the architecture.

L^2 regularization, three values of λ were tested: 0.001, 0.005 and 0.01. For both methods, the value that yielded better results was 0.01. Higher values for λ were also tested, which led to underfitting due to the aggressive shrinkage of coefficients to zero. In the case of dropout, a dropout layer was added after each hidden layer of the network, which randomly dropped 20% of the neurons of that layer. Figure 6.15 shows the comparison of the results obtained for all three methods. It should be pointed that the number of training epochs had to be increased due to the larger training times caused by the application of regularization methods.

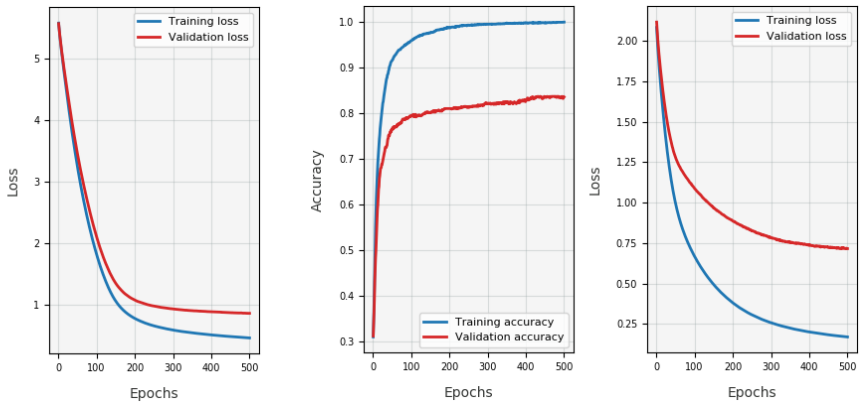
In this case, the validation accuracies obtained for each method were measured in the point where overfitting starts to take place, which led to the values of 80.6%, 83.6% and 81.1% for L^1 regularization, L^2 regularization and dropout, respectively. It can be concluded that L^2 regularization yielded the best results after training for 500 epochs. Afterwards, a final model was trained for 500 epochs on the entirety of the training data, using the architecture and fine-tuned hyper-parameters determined previously, which was evaluated on the test set only once, yielding an accuracy on the test set of 81.8%. The test accuracies obtained for all six subjects through the repetition of a similar model training process for each subject are presented in figure 6.16.

6.7.2 Group Division

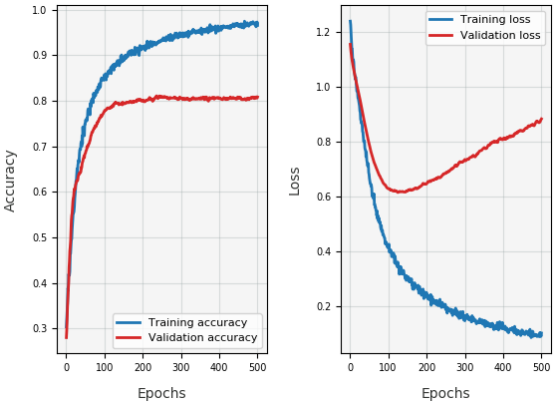
Regarding the division in groups for the second experimental session, the ITQ results were analysed and a score was given to each subject based on the answers to the questionnaire, where higher scores are indicative of subjects with an increased tendency to be immersed in the performed tasks. The ITQ comprises 18 questions in a *Likert* scale form of 7 values that are divided in four *subscales*, as the author calls them: *Focus*, *Involvement*, *Emotions*, and *Games*. Following the author's scoring



(a) L^1 regularization with $\lambda = 0.01$



(b) L^2 regularization with $\lambda = 0.01$



(c) Dropout of 20%

Figure 6.15: Training and validation accuracy and loss after applying three distinct regularization methods.

recommendations, each subscale's score was determined as the sum of the answers to the questions that regard that subscale. The subject's total score was then given by the sum of the subscale scores. Table 6.1 presents the score of each subscale and total score for each of the six subjects.

A set containing all the scores (*M* = 74.167, *SD* = 10.496) was then partitioned into two subsets, where the sum of each subset's scores was as close as possible, in order to generate two balanced groups that contain subjects with both high and low ITQ scores. The optimal group division led to the following groups:

- **Users 3, 5 and 6** (Sum of scores = 222)
- **Users 1, 2 and 4** (Sum of scores = 223)

The first group (users 3, 5 and 6) was chosen to test the classic GUI approach, while the second group (users 1, 2 and 4) was chosen to test the PAUI approach. The reasoning that led to this choice

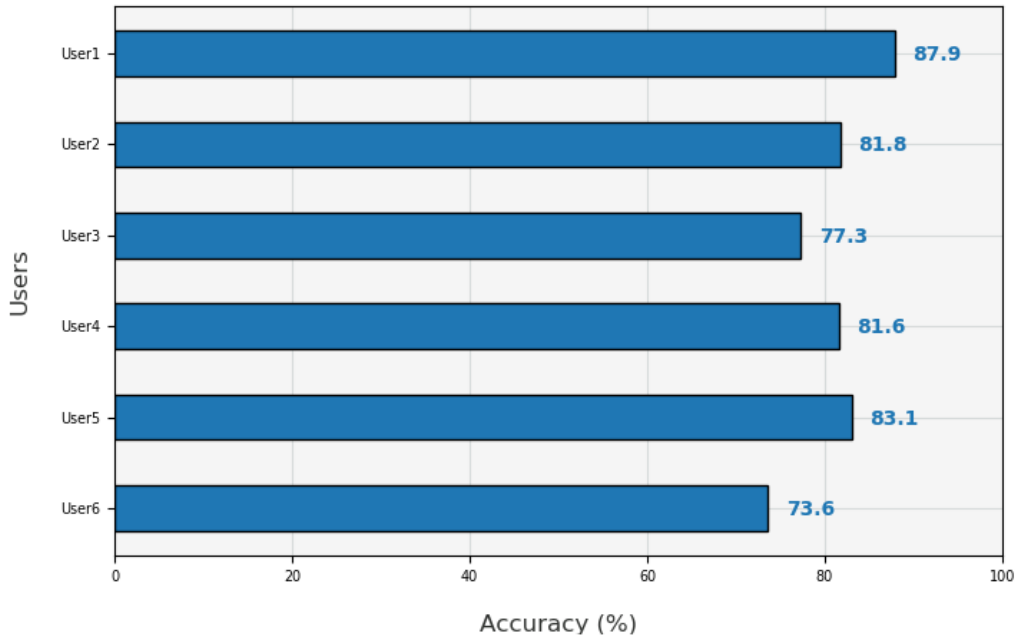


Figure 6.16: Accuracies obtained on the test set evaluation for all six subjects.

was that the second group's emotional state classifiers had a higher accuracy on average (83.8%) than the first group's (78.0%). Since the focus of this dissertation is to gain insight on the advantages of using a PAUI against the usage of a traditional GUI approach, choosing the group with higher classification accuracy to test the PAUI approach could potentially lead to a better representation of what the PAUI system can achieve, since the models have the potential to classify the correct emotional state more often.

6.7.3 Real-Time Classification

As mentioned previously, the classification accuracy obtained in the test set was measured for each participant and its results presented in figure 6.16, which show an average value of 80.9%. When the models were tested in real-time during the second session, the classification performance dropped significantly for each participant, to an average of 50.3% (a discussion on the decrease of the classification performance is presented in section 6.8). The accuracies obtained for each participant in both sessions are compared in figure 6.17. It should also be noted that these accuracies were obtained by considering equal true labels during each task (e.g. the whole stress task was labelled as stress).

6.7.4 Statistical Tests

As described in section 6.4, the evaluation experiment led to the acquisition of information relative to the task performance of each subject (completion time, number of objects found and the relative change

User ID	Focus	Involvement	Emotions	Games	Total
1	25	20	23	4	72
2	25	19	19	9	72
3	28	23	19	17	87
4	28	27	20	4	79
5	26	17	10	3	56
6	31	25	15	8	79

Table 6.1: ITQ total and individual subscale scores for each subject.

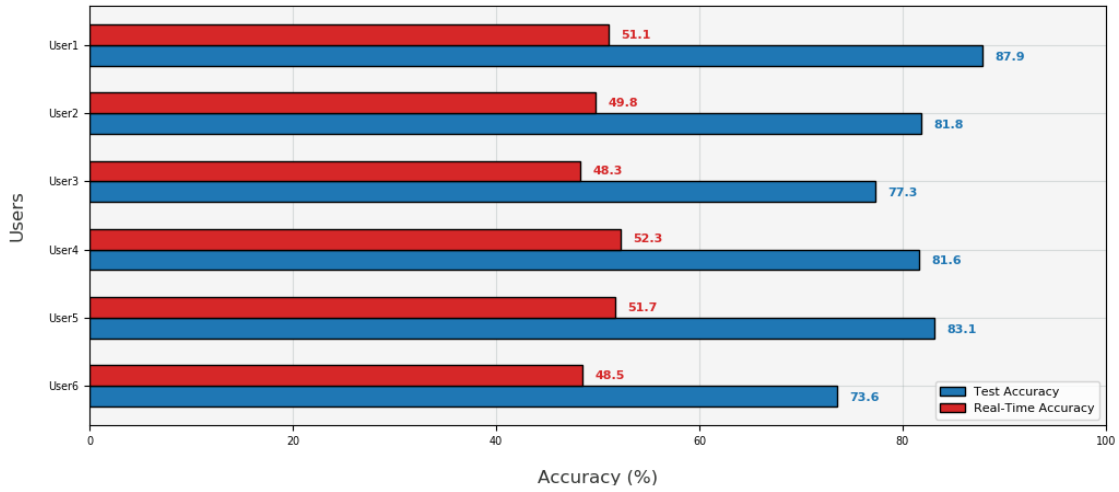


Figure 6.17: Comparison of the classification accuracies obtained for all six subjects between the test set and the evaluation experimental session.

of engagement levels), along with user experience information from the questionnaires filled after the execution of the experiment.

With the aim of analysing the gathered data on the task performance obtained under both approaches, the Shapiro-Wilk test was employed to check data for normality. In this test, the null hypothesis, H_0, is that the population is normally distributed. In order to determine the statistical significance of the test, a cutoff value is decided by the author, called *alpha*, which corresponds to the chance of obtaining a result similar to the one observed if H_0 was true. Alpha usually takes on a value of 0.05, meaning that if the *p-value* is less than 0.05, the null hypothesis can be rejected, meaning the data does not come from a normally distributed population. On the other hand, a p-value greater than 0.05 points that the population is normally distributed.

Since the samples collected over the course of the evaluation session are independent, given that

each subject only tested one of the approaches, the normality test can lead to the execution of two different tests to validate the statistical significance of each hypothesis: an *independent t-test* in case the data is normally distributed; a *Mann-Whitney U test* in case the data is not normally distributed. In the independent t-test, the null-hypothesis is that the population means of two unrelated groups are equal, while the alternative hypothesis is that the population means are different. For the Mann-Whitney U test, the null-hypothesis states that the distributions of both groups are equal, while the alternative hypothesis states the opposite. In both tests, the results are said to be statistically significant if the p-value is lower than alpha, since a low p-value indicates decreased support for the null hypothesis.

The statistical tests needed for the evaluation of this system were made using *IBM SPSS Statistics*, which license was provided by Instituto Superior Técnico. The results obtained from the execution of the tests are presented below (see appendix C for more details):

- **Completion Time**

Running a Shapiro-Wilk test for the completion time of the stress task obtained by each participant, a value of p = 0.945 was obtained, indicating that the completion time data was normally distributed. Additionally, it was shown that there was homogeneity of variances in the data, as assessed by the Levene's Test for Equality of Variances. Subsequently, an independent t-test was employed, which showed that there were no statistically significant differences in task completion time using the PAUI approach or the GUI approach (t(4) = -0.541, p = 0.617).

- **Objects Found**

Since the number of objects found by each participant during the workload task is a discrete value, it is assumed that the distribution that generates the data is not normal. As such, a Mann-Whitney U test was employed, which showed that there were no statistically significant differences in number of objects found by participants using the PAUI approach or the GUI approach (U = 3.5, p = 0.658).

- **Engagement Level**

Running a Shapiro-Wilk test for the relative change of the engagement levels between the rest task and the stress task for each participant, a value of p = 0.078 was obtained, indicating that the relative changes of engagement levels were drawn from a normal distribution. Additionally, it was shown that there was homogeneity of variances in the data, as assessed by the Levene's Test for Equality of Variances. Subsequently, an independent t-test was employed, which showed that there were no statistically significant differences in the relative change of the engagement levels using the PAUI approach or the GUI approach (t(4) = 1.504, p = 0.207).

- **User Experience**

When it comes to user experience, the USE Questionnaire was formed by a set of statements to be answered in a 7-point Likert scale. As Likert scales are of ordinal nature, they cannot be drawn from a normal distribution, and as such, no normality test was employed. The results associated with each question are presented in table 6.2, which shows the median and Inter-Quartile Range (IQR) obtained for each group in every question. The execution of a Mann-Whitney U test showed a tendency towards statistical significance on the sensation of effectiveness (Q1) perceived by the participants (U = 0.5, p = 0.072), as well as the ease of use (Q3) of the interface (U = 0.5, p = 0.077), where both elements have shown an improvement in the PAUI approach, when compared to the GUI approach. All the other questions led to statistically insignificant differences.

Additionally, the results from the DEQ revealed that users reported strong feelings associated with the relaxation subscale during the rest task in detriment of the emotions belonging to the anxiety and anger subscales, while obtaining higher scores in the anxiety subscale during the stress and workload tasks, in comparison to the relaxation subscale. The scores obtained in the anxiety subscale were slightly lower for the workload task, in relation to the stress task. The anger subscale obtained relatively low scores over the course of all three tasks. For information on the median and IQR obtained in each question, refer to tables C.1, C.2 and C.3 in appendix C.

Furthermore, all three participants that tested the PAUI approach reported feeling approximately the same emotions in both experimental sessions and observed that, even though the system could show more robustness (in terms of classification accuracy), the changes performed by the interface were helpful and eased the performance of the tasks.

6.8 Discussion

In order to evaluate the initial hypotheses presented in section 1.2, the user study described previously was conducted. The study allowed the gathering of data relative to the completion time, number of objects found, and relative changes of the engagement levels of each user. The USE Questionnaire was also used to draw conclusions on the usefulness, ease of use and satisfaction felt by the participants that tested each approach.

The results presented in the previous section showed a significant drop in terms of performance of the emotional state classifiers in comparison to the accuracies obtained in the test set, from 80.9% to 50.3%. The decline in the classification accuracy can be explained due to the inherently high variability present in physiological signals not only between subjects, as stated previously in section 6.7.1, but also within each subject. This means that mood fluctuations and different dispositions could have caused a difference in measured physiological signals for a certain emotional state, despite participants reporting

Question	PAUI	GUI
Usefulness		
Q1. The interface helped me be more effective.*	6 (0.5)	3 (1.5)
Q2. The interface made the things I wanted to accomplish easier to get done.	6 (0.5)	2 (2.0)
Ease of Use		
Q3. The interface is easy to use.*	6 (1.0)	4 (1.0)
Q4. The interface is user friendly.	6 (1.0)	5 (1.0)
Q5. I could recover from mistakes quickly and easily.	3 (0.5)	4 (0.5)
Satisfaction		
Q6. I am satisfied with the interface.	6 (1.0)	5 (1.0)
Q7. The interface works the way I want it to work.	6 (0.5)	3 (2.0)

Table 6.2: Results of the USE Questionnaire for both PAUI and GUI conditions. * indicates a tendency towards statistical significance. The values shown are the median and the Inter-Quartile Range (IQR) in parentheses.

identical emotions between both sessions. The variability within subjects was also visualized clearly with the application of a t-SNE to a dataset containing data from both sessions of a single user, which can be observed in figures C.11, C.12 and C.13 of appendix C. The intrinsic variability of these signals is indicative of the necessity of a larger dataset, possibly containing data from different days, in order to accurately capture the nuances of an underlying emotional state. Additionally, physiological signals have low signal-to-noise ratios, which led the trained models to recognize patterns in noisy signals that could have little meaning regarding the emotional state of the user. Moreover, the knowledge of the correct class label to attribute to each data point is not obvious for emotion classification problems, meaning that some data points were likely to be incorrectly labelled and thus contributed to a reduction of the model's generalization power.

Regarding the task performance metrics, the results obtained from the statistical tests were not conclusive, since no statistical significance was found for the tests regarding the completion times, the number of objects found and the relative changes of the engagement levels. Still, the mean completion time of the stress task measured for users of the PAUI approach has shown a slight reduction over the one measured for users of the GUI approach, while the number of objects found was mildly higher for the PAUI approach. These small differences between the two approaches can be explained by the poor performance of the emotional state classifier, which caused a limited experience of the full capabilities of the system by the PAUI users. In terms of the relative change of engagement levels, the results

showed a higher relative change of the engagement levels in users of the PAUI approach in comparison with users of the GUI approach, which could be indicative that the attentive properties of the system led to higher levels of engagement. Nonetheless, the small number of subjects that were available for the experimental sessions makes it difficult to draw conclusions regarding hypotheses **H1**, **H2** and **H3**.

When it comes to the USE Questionnaire, a tendency towards statistical significance was achieved for questions Q1 and Q3, which assess the feeling of effectiveness and the sensation of ease of use felt by the participants, respectively. In both cases, the PAUI approach led to an improvement of these sentiments over the GUI approach. Despite the questionnaires being a subjective form of validation, the results go in accordance with the attentive interface changes, especially for question Q1, since the PAUI focuses on enhancing the effectiveness of the users by presenting information on the interface in a clear and easy way to understand in moments of cognitive workload, which does not happen in the GUI. Nonetheless, it should be pointed that user reported feelings can be slightly different from the real feelings, which can make the results somewhat unreliable, especially in experiments with a small number of participants such as the one presented. Additionally, even though the attentive strategies adopted focused on improving the user's interaction with the interface, it is also possible for users to have a biased preference for the PAUI simply due to the fact that it is dynamic. Even though a tendency towards users of the PAUI exhibiting an improved sense of effectiveness over users of the GUI was verified, the lack of statistical significance cannot assure the validation of the hypothesis **H4**.

Although not statistically significant, the small improvements obtained in the task performance metrics are indicative that more significant differences could be achieved with the aid of a more powerful classifier, preferably trained with a larger and more refined dataset collected over the course of a longer teleoperation session. Moreover, the lack of statistical significance was expected due to the very small number of subjects available for the experiment (only 3 subjects for each condition). This presented a major limitation in the results obtained, which can be improved in future studies that rely on a larger number of subjects. Furthermore, the design of the experiment proved to be effective in replicating the pretended emotional states on all the participants. The tasks designed for the user evaluation, which were presented in section 6.3, successfully induced the feelings of relaxation (in the rest task) or anxiety (in the stress and workload tasks), as reported by participants in the Discrete Emotions Questionnaire. These results go in accordance with the strategies adopted for each task. The lack of activity during the rest task led to a restful state, possibly with windows of boredom, while the stress task offered a much more challenging environment through the usage of a timer, sound beeps and obstacles. The workload task also triggered a high cognitive overload state by requiring users to divide their attention between finding objects and answering questions. The absence of a timer and the usage of a less challenging environment could explain the decrease in the levels of anxiety reported by participants, when compared to the stress task. However, the questionnaire also reflects the difficulty in differentiating the

sensations felt between the stress and workload tasks, as the questionnaire did not provide subscales that could differentiate them, which makes sense given that both states can cause a similar feeling of cognitive overload on the users. Nevertheless, the sensations reported by the participants clearly show the effectiveness of the emotion induction tasks, which can be used in future studies that use both the USAR simulator or the real robot. Additionally, all three participants that tested the PAUI approach were able to experience moments where the classifier was predicting the correct emotional state. Despite the inconsistency caused by the poor classification performance, the PAUI users reported that the changes performed by the PAUI were helpful and allowed a better understanding of the environment, which indicates that the attentive measures adopted by the system can lead to the improvement of the user's focus in difficult situations. Another consideration is that, even though the PAUI is to be used by experts, the users that were available for the evaluation of the system were not experts. This can have an impact on the perceived usefulness of the attentive strategies of the interface, since expert users might miss some of the interface elements in redefined formats such as the one applied in the workload state. Ultimately, it would be of interest if the system's attentive measures could be customized by users, as it would solve the problem of the changes being useful for some users but not for others.

Summing up, the small number of subjects available and the poor classification performance were the main limitations of the results obtained, leading to their statistical insignificance. The success of both the emotion induction strategies adopted and the attentive measures of the PAUI is indicative of potential improvements in future works that comprehend a significant number of subjects, along with enhanced classifiers. However, due to the lack of significance found in the statistical tests carried out for this study, the initial research statement cannot be validated.

7

Conclusion

In today's society, user's attention is proving the be the most valuable resource in the performance of user interfaces applied both in the HCI and the HRI fields. This thesis addressed the problem of managing user's attention in the field of robot teleoperation, particularly in USAR environments, where it is common for user interfaces to be very complex and exhibit displays of information that can cause cognitive overload on their operators, potentially compromising the success of the missions. In order to achieve this goal, a prototype of a PAUI applied to the robot teleoperation field was developed, combining the classification of emotional states in real-time with the design of attentive strategies that converted the original interface into an attentive user interface without access to its source code. The solution developed is generic and can be applied to other case studies, although some work is required for the adaptation of the system to the chosen interface. In this dissertation, the PAUI was applied to the RAPOSA's case study.

Through the execution of a user study with 6 people, it was possible to analyze quantitative and qualitative measures of the user's task performance and preference towards the usage of a physiologically attentive system in comparison to the classic interface approach. The study carried out did not find any significant differences in terms of task performance between both groups, although there was a tendency for an improvement in the feeling of effectiveness and ease of usage, as reported by users in the questionnaires. Hence, the initial research statement could not be validated.

The major contribution of this thesis was the development of the first prototype of the PAUI applied to the pre-existing RAPOSA's interface, where a great emphasis was placed on creating effective methods of managing user's attention in moments of cognitive workload, taking into account not only the concerns expressed by previous research in the design of attentive user interfaces, but also the problems that are recurrent in interfaces developed for USAR robots. The planning of tasks designed to induce specific emotional states (rest, stress and workload) was also an important step in realizing what types of strategies can be implemented in re-creating real life USAR environments and successfully inducing the sensations felt by operators when going through these situations. The development of a USAR simulator also contributed largely to the success of the previous point, by allowing the modelling of custom 3D environments and the implementation of a large set of features that simulate the teleoperation process of the real robot, thus producing a faithful representation of the reality that can be used in the future studies.

Furthermore, the results obtained considering the small number of subjects available, allied with the limited capabilities of the emotional state classifier are indicative of potential improvements in future work. A major element that needs specialized attention is the robustness of the emotional state classifier. The classification of emotions is by itself a challenging task, and the results clearly show the need of obtaining a larger dataset that can accurately represent the underlying distribution that generates physiological data. The classifier would also benefit largely from an investment in feature engineering

methods, which can use knowledge about physiological signals to expose the algorithm to the most valuable information present in the extracted features, leading to an improvement in the robustness of the classifier. Additionally, new strategies for managing user attention can be implemented by giving emphasis to the eye tracking device. The use of gaze tracing can give insight into the regions of the interface where users tend to stare at the most, which can be advantageous in providing new ways for the interface to attend to the user's needs. Moreover, it could be beneficial to ease the process of adapting the system to other case studies in order to increase the flexibility of the solution. Finally, even though the simulator is a great tool for the evaluation of systems similar as the one presented here, the execution of future evaluation studies with the real robot could be helpful in providing users an improved understanding of the robot teleoperation process, allowing the employment of attentive measures that offer a greater contribution to the task performance achieved.

Bibliography

[1] B. Safaei, A. M. H. Monazzah, M. B. Bafroei, and A. Ejlali, "Reliability side-effects in Internet of Things application layer protocols," in *2017 2nd International Conference on System Reliability and Safety (ICSRS)*. IEEE, 2017, pp. 207–212.

[2] A. Bulling, "Pervasive Attentive User Interfaces," *IEEE Computer*, vol. 49, no. 1, pp. 94–98, 2016.

[3] R. Vertegaal *et al.*, "Attentive user interfaces," *Communications of the ACM*, vol. 46, no. 3, pp. 30–33, 2003.

[4] A. C. Dirican and M. Göktürk, "Psychophysiological measures of human cognitive states applied in human computer interaction," *Procedia Computer Science*, vol. 3, pp. 1361–1367, 2011.

[5] D. Chen, *The Physiologically Attentive User Interface Towards A Physiological Model of Interruptability*. Citeseer, 2006.

[6] G. Singh, S. Bermúdez i Badia, R. Ventura, and J. L. Silva, "Physiologically attentive user interface for robot teleoperation: real time emotional state estimation and interface modification using physiology, facial expressions and eye movements," in *11th International Joint Conference on Biomedical Engineering Systems and Technologies*. SCITEPRESS-Science and Technology Publications, 2018, pp. 294–302.

[7] ISO (2018) Ergonomics of human-system interaction - Part 11: Usability: Definitions and concepts (ISO/DIS 9241-11:2018).

[8] R. Vertegaal, J. S. Shell, D. Chen, and A. Mamuji, "Designing for augmented attention: Towards a framework for attentive user interfaces," *Computers in Human Behavior*, vol. 22, no. 4, pp. 771–789, 2006.

[9] D. Garlan, D. P. Siewiorek, A. Smailagic, and P. Steenkiste, "Project Aura: Toward distraction-free pervasive computing," *IEEE Pervasive computing*, vol. 1, no. 2, pp. 22–31, 2002.

[10] J. Oliver and B. García, “Innovations in User Interfaces for Pervasive Computing,” in *UBICOMM 2013-7th International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies*, 2013.

[11] D. Chen and R. Vertegaal, “Using mental load for managing interruptions in physiologically attentive user interfaces,” in *CHI’04 Extended Abstracts on Human Factors in Computing Systems*, 2004, pp. 1513–1516.

[12] C. Guo and E. Sharlin, “Exploring the use of tangible user interfaces for human-robot interaction: a comparative study,” in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2008, pp. 121–130.

[13] J. R. Millan, F. Renkens, J. Mourino, and W. Gerstner, “Noninvasive brain-actuated control of a mobile robot by human EEG,” *IEEE Transactions on Biomedical Engineering*, vol. 51, no. 6, pp. 1026–1033, 2004.

[14] B. Zhang, J. Wang, and T. Fuhlbrigge, “A review of the commercial brain-computer interface technology from perspective of industrial robotics,” in *2010 IEEE International Conference on Automation and Logistics*. IEEE, 2010, pp. 379–384.

[15] M. Baker, R. Casey, B. Keyes, and H. A. Yanco, “Improved interfaces for human-robot interaction in urban search and rescue,” in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, vol. 3. IEEE, 2004, pp. 2960–2965.

[16] J. M. Riley and M. R. Endsley, “The hunt for situation awareness: Human-robot interaction in search and rescue,” in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 48, no. 3. SAGE Publications Sage CA: Los Angeles, CA, 2004, pp. 693–697.

[17] K. Anderson and P. W. McOwan, “A real-time automated system for the recognition of human facial expressions,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 36, no. 1, pp. 96–105, 2006.

[18] I. Hupont, S. Baldassarri, and E. Cerezo, “Facial emotional classification: from a discrete perspective to a continuous emotional space,” *Pattern Analysis and Applications*, vol. 16, no. 1, pp. 41–54, 2013.

[19] F. Agrafioti, D. Hatzinakos, and A. K. Anderson, “ECG pattern analysis for emotion detection,” *IEEE Transactions on Affective Computing*, vol. 3, no. 1, pp. 102–115, 2011.

[20] J. Wagner, J. Kim, and E. André, “From physiological signals to emotions: Implementing and comparing selected methods for feature extraction and classification,” in *2005 IEEE International Conference on Multimedia and Expo*. IEEE, 2005, pp. 940–943.

[21] K. H. Kim, S. W. Bang, and S. R. Kim, “Emotion recognition system using short-term monitoring of physiological signals,” *Medical and Biological Engineering and Computing*, vol. 42, no. 3, pp. 419–427, 2004.

[22] P. C. Petrantonakis and L. J. Hadjileontiadis, “A novel emotion elicitation index using frontal brain asymmetry for enhanced EEG-based emotion recognition,” *IEEE Transactions on Information Technology in Biomedicine*, vol. 15, no. 5, pp. 737–746, 2011.

[23] X.-W. Wang, D. Nie, and B.-L. Lu, “Emotional state classification from EEG data using machine learning approach,” *Neurocomputing*, vol. 129, pp. 94–106, 2014.

[24] W.-L. Zheng, J.-Y. Zhu, Y. Peng, and B.-L. Lu, “EEG-based emotion classification using deep belief networks,” in *2014 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2014, pp. 1–6.

[25] C. Marques, J. Cristóvao, P. Alvito, P. Lima, J. Frazao, I. Ribeiro, and R. Ventura, “A search and rescue robot with tele-operated tether docking system,” *Industrial Robot: An International Journal*, vol. 34, no. 4, pp. 332–338, 2007.

[26] D. McDuff, A. Mahmoud, M. Mavadati, M. Amr, J. Turcot, and R. e. Kaliouby, “AFFDEX SDK: a cross-platform real-time multi-face expression recognition toolkit,” in *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, 2016, pp. 3723–3726.

[27] M. A. Nielsen, *Neural Networks and Deep Learning*. Determination Press, 2015.

[28] S. Ruder, “An overview of gradient descent optimization algorithms,” *arXiv preprint arXiv:1609.04747*, 2016.

[29] F. Chollet, *Deep Learning with Python*. Manning Publications, 2017.

[30] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[31] D. Anguita, L. Ghelardoni, A. Ghio, L. Oneto, and S. Ridella, “The ’K’ in K-fold Cross Validation,” in *ESANN*, 2012.

[32] K. P. Murphy, *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[33] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[34] A. Géron, *Hands-On Machine Learning with Scikit-Learn & TensorFlow*. O’Reilly Media, 2017.

[35] J. Kukačka, V. Golkov, and D. Cremers, “Regularization for deep learning: A taxonomy,” *arXiv preprint arXiv:1710.10686*, 2017.

[36] P. Baldi and R. Vershynin, “The capacity of feedforward neural networks,” *Neural Networks*, vol. 116, pp. 288–311, 2019.

[37] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[38] J. Shlens, “A Tutorial on Principal Component Analysis,” *arXiv preprint arXiv:1404.1100*, 2014.

[39] L. v. d. Maaten and G. Hinton, “Visualizing data using t-SNE,” *Journal of Machine Learning Research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[40] B. A. Forouzan, *Data Communications and Networking*. McGraw Hill, 2007.

[41] B. G. Witmer and M. J. Singer, “Measuring presence in virtual environments: A presence questionnaire,” *Presence*, vol. 7, no. 3, pp. 225–240, 1998.

[42] A. Němcová, L. Maršánová, and R. Smíšek, “Recommendations for ECG acquisition using BITalino,” in *Conference Paper EEICT*, vol. 1, 2016, pp. 543–47.

[43] T. McMahan, I. Parberry, and T. D. Parsons, “Evaluating player task engagement and arousal using electroencephalography,” *Procedia Manufacturing*, vol. 3, pp. 2303–2310, 2015.

[44] A. M. Lund, “Measuring usability with the USE questionnaire,” *Usability Interface*, vol. 8, no. 2, pp. 3–6, 2001.

[45] C. Harmon-Jones, B. Bastian, and E. Harmon-Jones, “The Discrete Emotions Questionnaire: A New Tool for Measuring State Self-reported Emotions,” *PloS one*, vol. 11, no. 8, p. e0159915, 2016.

[46] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems*, 2012, pp. 1097–1105.

[47] Y. Bengio, “Practical recommendations for gradient-based training of deep architectures,” in *Neural Networks: Tricks of the Trade*. Springer, 2012, pp. 437–478.

Workload Questions and Problems

1st Session – Part 1

Question Number	Question	Answer
1	**14 * 4**	**56**
2	**35 + 62**	**97**
3	**What is the value read on the interface for: PC Battery**	**N/A**
4	**18 * 3**	**54**
5	**75 – 28**	**47**
6	**27 + 35**	**62**
7	**42 * 4**	**168**
8	**How many red cubes have you found so far?**	**N/A**
9	**Find the missing number in the series: 4 8 12 20 __ (1)**	**32**
10	**78 + 64**	**142**
11	**What is the value read on the interface for: CO**	**N/A**
12	**16 * 3**	**48**
13	**36 / 4**	**9**
14	**23 * 4**	**92**
15	**29 + 82**	**111**
16	**36 / 4**	**9**
17	**Find the missing number in the series: 4 8 16 32 __ (2)**	**64**
18	**What is the value read on the interface for: Humidity**	**N/A**
19	**What objects do you see around you? Which room do you think you are in?**	**N/A**
20	**46 – 28**	**18**
21	**26 + 35**	**61**
22	**56 / 8**	**7**
23	**What is the value read on the interface for: Linear Velocity**	**N/A**
24	**92 – 75**	**17**
25	**Find the missing number in the series: 1 4 9 __ 25 (3)**	**16**
26	**93 / 3**	**31**
27	**Where was the last red cube you found?**	**N/A**
28	**Which symbol in the answer figure completes the sequence in the problem figure? (9)**	**C**
29	**81 - 65**	**16**
30	**What is the value read on the interface for: Angular Velocity**	**N/A**
31	**160 / 4**	**40**
32	**89 + 27**	**116**
33	**72 / 3**	**24**
34	**How many red cubes have you found so far?**	**N/A**
35	**76 - 54**	**22**
36	**51 + 63**	**114**

Figure A.1: Set of questions asked during the first iteration of the workload task in the first experimental session.

1st Session – Part 2

Question Number	Question	Answer
1	**13 * 5**	**65**
2	**48 / 4**	**12**
3	**What is the value read on the interface for: Motor Battery**	**N/A**
4	**74 + 26**	**100**
5	**5 * 99**	**495**
6	**76 - 58**	**18**
7	**26 * 4**	**104**
8	**How many red cubes have you found so far?**	**N/A**
9	**Find the missing number in the series: 7 5 8 4 9 3 10 __ (5)**	**2**
10	**65 / 5**	**13**
11	**What is the value read on the interface for: HS**	**N/A**
12	**49 + 45**	**94**
13	**4 * 58**	**232**
14	**72 / 2**	**36**
15	**86 - 39**	**47**
16	**34 * 4**	**136**
17	**Find the missing number in the series: 8723 3872 2387 __ (6)**	**7238**
18	**What is the value read on the interface for: Inf Gas 1**	**N/A**
19	**What objects do you see around you? Which room do you think you are in?**	**N/A**
20	**45 / 5**	**9**
21	**104 – 38**	**66**
22	**41 + 85**	**126**
23	**What is the value read on the interface for: Pitch**	**N/A**
24	**91 – 44**	**47**
25	**Find the missing number in the series: 99 92 86 81 __ 74 (7)**	**77**
26	**5 * 21**	**105**
27	**Where was the last red cube you found?**	**N/A**
28	**Which of the two figures are identical? (10)**	**C, E**
29	**35 / 5**	**7**
30	**What is the value read on the interface for: Inf Gas 2**	**N/A**
31	**81 + 42**	**123**
32	**360 / 8**	**45**
33	**89 – 13**	**76**
34	**How many red cubes have you found so far?**	**N/A**
35	**4 * 38**	**152**
36	**108 / 3**	**36**
37	**What is the value read on the interface for: Roll**	**N/A**
38	**89 + 39**	**128**
39	**Find the missing number in the series: 1 2 6 24 120 __ (8)**	**720**
40	**87 – 63**	**24**

Figure A.2: Set of questions asked during the second iteration of the workload task in the first experimental session.

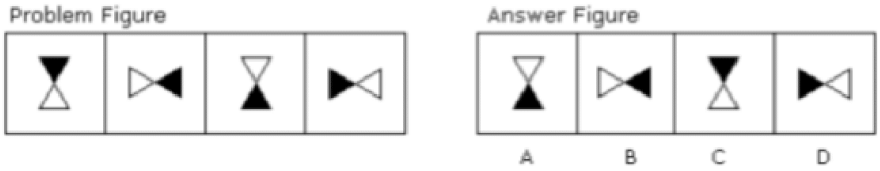
2nd Session

Question Number	Question	Answer
1	**27 * 3**	**81**
2	**96 / 4**	**24**
3	**What is the value read on the interface for: Linear Velocity**	**N/A**
4	**41 + 37**	**78**
5	**65 - 46**	**19**
6	**81 - 59**	**22**
7	**54 * 5**	**270**
8	**How many red cubes have you found so far?**	**N/A**
9	**Find the missing number in the series: 3 6 12 24 __ 96 (1)**	**48**
10	**72 / 4**	**18**
11	**What is the value read on the interface for: Humidity**	**N/A**
12	**39 + 22**	**61**
13	**14 * 5**	**70**
14	**63 / 3**	**21**
15	**75 - 56**	**19**
16	**22 * 6**	**132**
17	**Find the missing number in the series: 54 47 41 36 __ 29 (2)**	**32**
18	**What is the value read on the interface for: CO**	**N/A**
19	**What objects do you see around you? Which room do you think you are in?**	**N/A**
20	**84 / 7**	**12**
21	**56 * 5**	**280**
22	**56 + 19**	**75**
23	**What is the value read on the interface for: HS**	**N/A**
24	**62 – 34**	**28**
25	**Find the missing number in the series: 7 23 9 21 11 __ 13 17 (3)**	**19**
26	**6 * 15**	**90**
27	**Where was the last red cube you found?**	**N/A**
28	**Which of the answer figures is a rotation of the question figure? (5)**	**C**
29	**52 / 2**	**26**
30	**What is the value read on the interface for: Inf Gas 1**	**N/A**
31	**39 + 65**	**104**
32	**210 / 7**	**30**
33	**81 - 49**	**32**
34	**How many red cubes have you found so far?**	**N/A**
35	**48 * 3**	**144**
36	**210 / 7**	**30**

Figure A.3: Set of questions asked during the workload task in the second experimental session.

Nr.	Series							
1	4		8		12		20	__
2	4		8		16		32	__
3	1		4		9		__	25
4	2		4		7		__	16
5	7	5	8	4	9	3	10	__
6	8723		3872		2387		__	
7	99	92	86	81	__	74		
8	1	2	6	24	120	__		

9.



10.

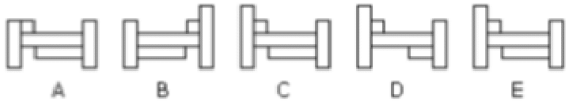


Figure A.4: Sequences of numbers and visual logic problems used in the first experimental session.

Nr.	Series							
1	3		6		12	24	__	96
2	54		47		41	36	__	29
3	7	23	9	21	11	__	13	17
4	87		76		67	60	55	__

5.

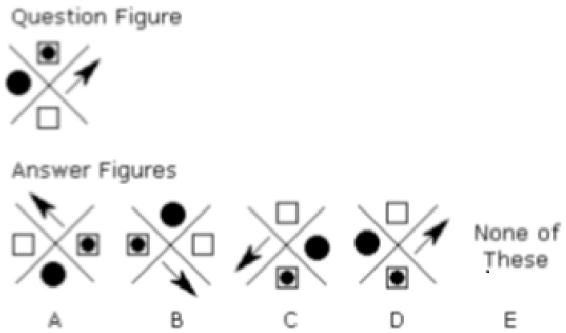


Figure A.5: Sequences of numbers and visual logic problems used in the second experimental session.

B

COVID-19 Safety Measures

Taking in consideration the recommendations issued by the Department of Safety, Health and Hygiene of Instituto Superior Técnico (IST), a set of guidelines were created to minimize the risk of infection and spread of the virus during the user evaluation sessions. The measures adopted can be consulted below:

- **Rigorous hand cleaning:** Before entering the experiment room, each user was required to wash their hands with water and soap for 20 seconds;
- **Mandatory use of mask:** After the hand cleaning procedures, each user was given a surgical mask to use in case they did not already possess their own;
- **Disinfection of hardware and surfaces:** Between each utilization by different users, all the hardware and surfaces of the setup were disinfected using 70% alcohol based wipes;
- **Avoidance of large gatherings:** A strict schedule was defined in order to guarantee that each user would only interact with the experiment supervisor, giving a 30 minute break between each subject that took in account any possible delays that might have happened.

- **Screen sharing supervision:** Since the context of the user study required the extraction of facial expressions, which would not be possible if the user wore a face mask, the supervisor only interacted with the subject for the purpose of placing the Bitalino electrodes and to give a contextualization of the experiment. After this point, the subject was left completely alone in the setup room, after which he/she was allowed to remove the face mask with care. After leaving, the supervisor monitored the experiment via the screen sharing tool integrated in Skype.

C

SPSS and Auxiliary Results

C.1 Normality Tests

Testes de Normalidade

	Kolmogorov-Smirnov[a]			Shapiro-Wilk		
	Estatística	df	Sig.	Estatística	df	Sig.
Time	,193	6	,200*	,979	6	,945
Objects	,214	6	,200*	,958	6	,804
Engagement	,269	6	,200*	,814	6	,078

*. Este é um limite inferior da significância verdadeira.

a. Correlação de Significância de Lilliefors

Figure C.1: Normality test results.

C.2 Completion Time

Estatísticas de grupo

	Group	N	Média	Erro Desvio	Erro padrão da média
Time	1,00	3	747,3333	55,71654	32,16796
	2,00	3	773,3333	61,85736	35,71337

Figure C.2: Group statistics for the completion time of the stress task. Group 1 and 2 refers to the PAUI and GUI groups, respectively.

Teste de amostras independentes

		Teste de Levene para igualdade de variâncias		teste-t para Igualdade de Médias						
									95% Intervalo de Confiança da Diferença	
		Z	Sig.	t	df	Sig. (2 extremidades)	Diferença média	Erro padrão de diferença	Inferior	Superior
Time	Variâncias iguais assumidas	,063	,813	-,541	4	,617	-26,00000	48,06477	-159,44920	107,44920
	Variâncias iguais não assumidas			-,541	3,957	,618	-26,00000	48,06477	-160,02230	108,02230

Figure C.3: Results of the independent t-test for the completion time of the stress task.

C.3 Objects Found

Estatísticas de grupo

	Group	N	Média	Erro Desvio	Erro padrão da média
Objects	1,00	3	6,0000	1,00000	,57735
	2,00	3	5,6667	2,08167	1,20185

Figure C.4: Group statistics for the number of objects found during the workload task. Group 1 and 2 refers to the PAUI and GUI groups, respectively.

Estatísticas de teste[a]

	Objects
U de Mann-Whitney	3,500
Wilcoxon W	9,500
Z	-,443
Significância Sig. (bilateral)	,658
Sig exata [2*(Sig. de 1 extremidade)]	,700[b]

Figure C.5: Results of the Mann-Whitney U test for the number of objects found during the workload task.

C.4 Engagement Levels

Estatísticas de grupo

	Group	N	Média	Erro Desvio	Erro padrão da média
Engagement	1,00	3	175,8400	149,26038	86,17552
	2,00	3	41,6133	40,37590	23,31104

Figure C.6: Group statistics for the relative changes of the engagement levels from the rest to the workload task. Group 1 and 2 refers to the PAUI and GUI groups, respectively.

Teste de amostras independentes

		Teste de Levene para igualdade de variâncias		teste-t para Igualdade de Médias						
									95% Intervalo de Confiança da Diferença	
		Z	Sig.	t	df	Sig. (2 extremidades)	Diferença média	Erro padrão de diferença	Inferior	Superior
Engagement	Variâncias iguais assumidas	6,620	,062	1,504	4	,207	134,22667	89,27276	-113,63424	382,08757
	Variâncias iguais não assumidas			1,504	2,291	,256	134,22667	89,27276	-206,70031	475,15364

Figure C.7: Results of the independent t-test for the relative changes of the engagement levels from the rest to the workload task.

C.5 USE Questionnaire

Estatísticas de teste[a]

	Q1	Q2	Q3	Q4	Q5	Q6	Q7
U de Mann-Whitney	,500	2,000	,500	1,000	3,000	3,500	2,000
Wilcoxon W	6,500	8,000	6,500	7,000	9,000	9,500	8,000
Z	-1,798	-1,179	-1,771	-1,623	-,745	-,471	-1,159
Significância Sig. (bilateral)	,072	,239	,077	,105	,456	,637	,246
Sig exata [2*(Sig. de 1 extremidade)]	,100[b]	,400[b]	,100[b]	,200[b]	,700[b]	,700[b]	,400[b]

a. Variável de Agrupamento: Group

b. Não corrigido para vínculos.

Figure C.8: Results of the Mann Whitney U test for the responses obtained in the USE Questionnaire.

C.6 DEQ Results

During the rest task, to what extent did you experience these emotions?	Median (IQR)
Subscale: Relaxation	
Calm	7.0 (0.0)
Relaxation	6.5 (1.0)
Easy going	6.0 (0.0)
Subscale: Anxiety	
Anxiety	1.0 (0.0)
Worry	1.0 (0.75)
Nervous	1.0 (0.75)
Subscale: Anger	
Anger	1.0 (0.0)
Rage	1.0 (0.0)
Mad	1.0 (0.0)

Table C.1: Results of the DEQ reported by all participants during the rest task.

During the stress task, to what extent did you experience these emotions?	Median (IQR)
Subscale: Relaxation	
Calm	1.5 (2.5)
Relaxation	1.5 (1.75)
Easy going	2.5 (1.75)
Subscale: Anxiety	
Anxiety	5.0 (1.5)
Worry	5.0 (0.75)
Nervous	4.5 (3.0)
Subscale: Anger	
Anger	1.0 (0.75)
Rage	1.0 (1.5)
Mad	1.5 (2.5)

Table C.2: Results of the DEQ reported by all participants during the stress task.

During the workload task, to what extent did you experience these emotions?	Median (IQR)
Subscale: Relaxation	
Calm	1.5 (1.75)
Relaxation	1.0 (0.75)
Easy going	2.5 (1.75)
Subscale: Anxiety	
Anxiety	4.5 (1.75)
Worry	5.0 (2.25)
Nervous	4.5 (1.75)
Subscale: Anger	
Anger	3.0 (0.75)
Rage	1.5 (3.25)
Mad	1.5 (3.25)

Table C.3: Results of the DEQ reported by all participants during the workload task.

C.7 t-SNE Visualizations

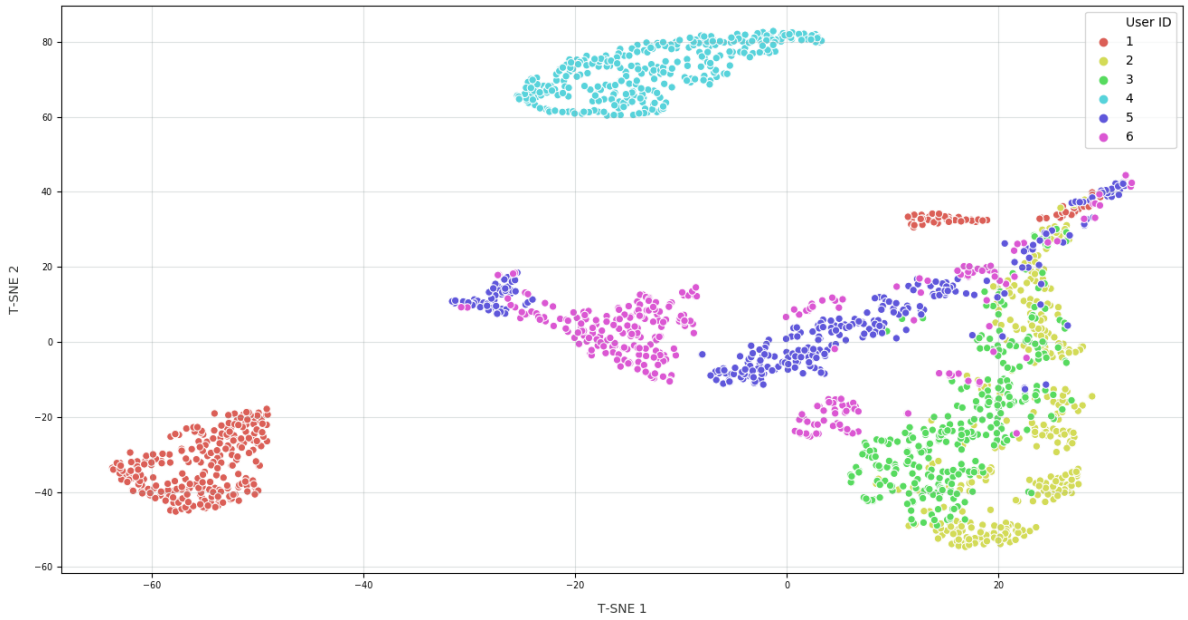


Figure C.9: Visualization of the two-dimensional embedded space that results from the application of t-SNE (Perplexity = 40, Number of iterations = 2000) to the data collected from each user during the rest task, where each user is represented by a different color.

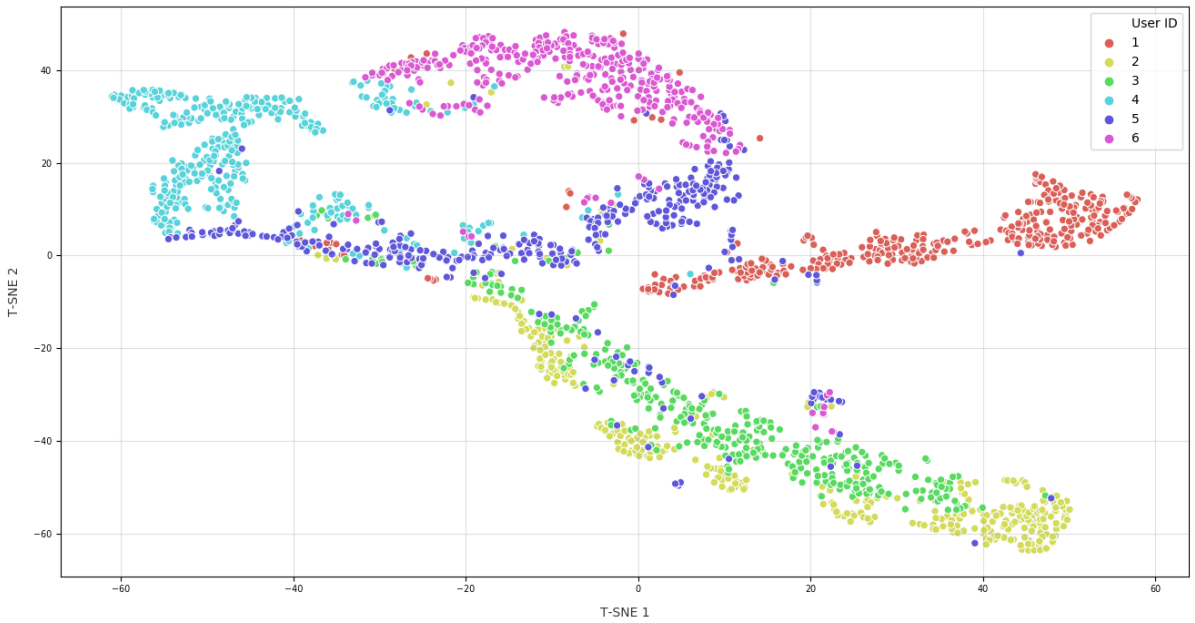


Figure C.10: Visualization of the two-dimensional embedded space that results from the application of t-SNE (Perplexity = 40, Number of iterations = 2000) to the data collected from each user during the workload task, where each user is represented by a different color.

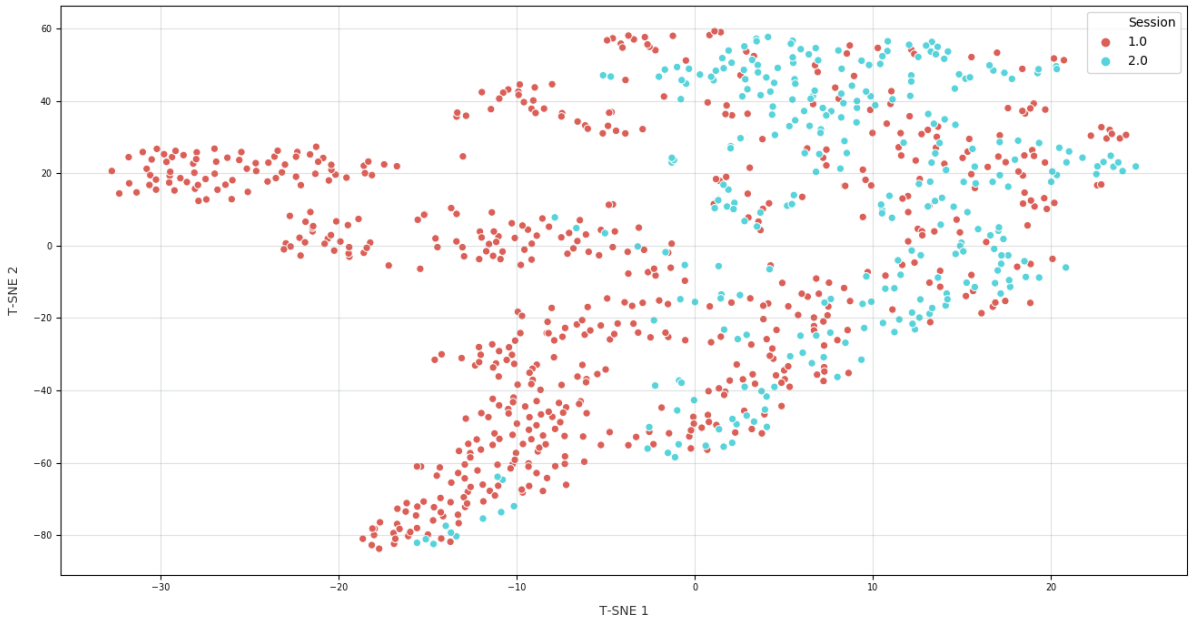


Figure C.11: Visualization of the two-dimensional embedded space that results from the application of t-SNE (Perplexity = 30, Number of iterations = 2000) to the data collected from a single user (user 2 in this case, although the results obtained were similar for all users) during the rest task in both experimental sessions, where each session is represented by a different color.

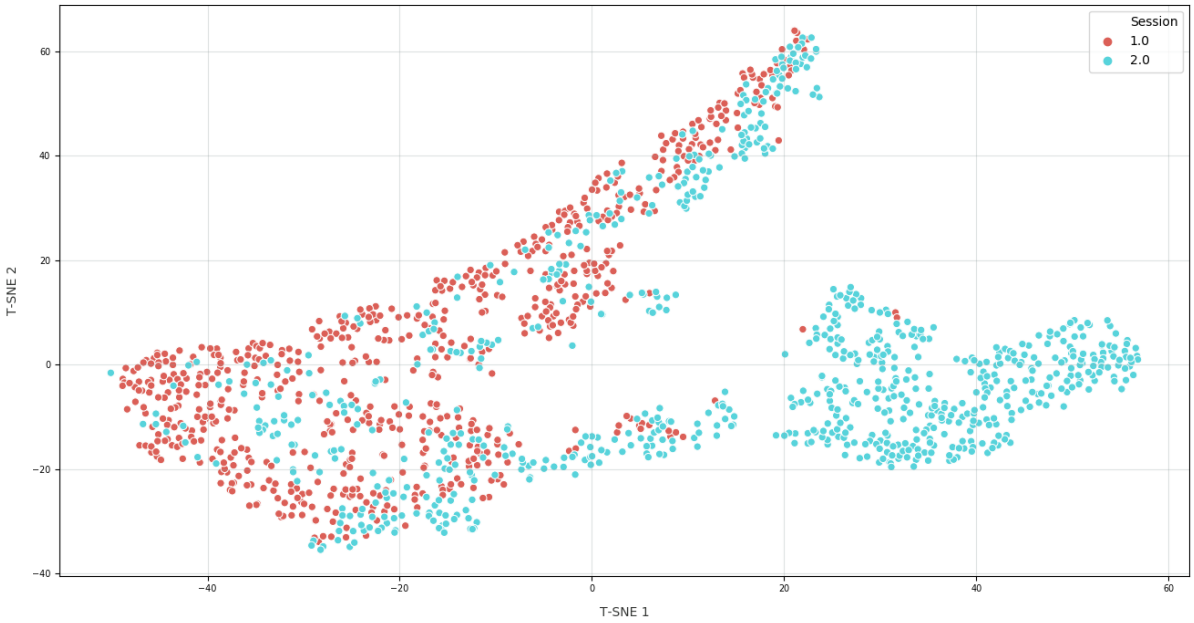


Figure C.12: Visualization of the two-dimensional embedded space that results from the application of t-SNE (Perplexity = 30, Number of iterations = 2000) to the data collected from a single user (user 2 in this case, although the results obtained were similar for all users) during the stress task in both experimental sessions, where each session is represented by a different color.

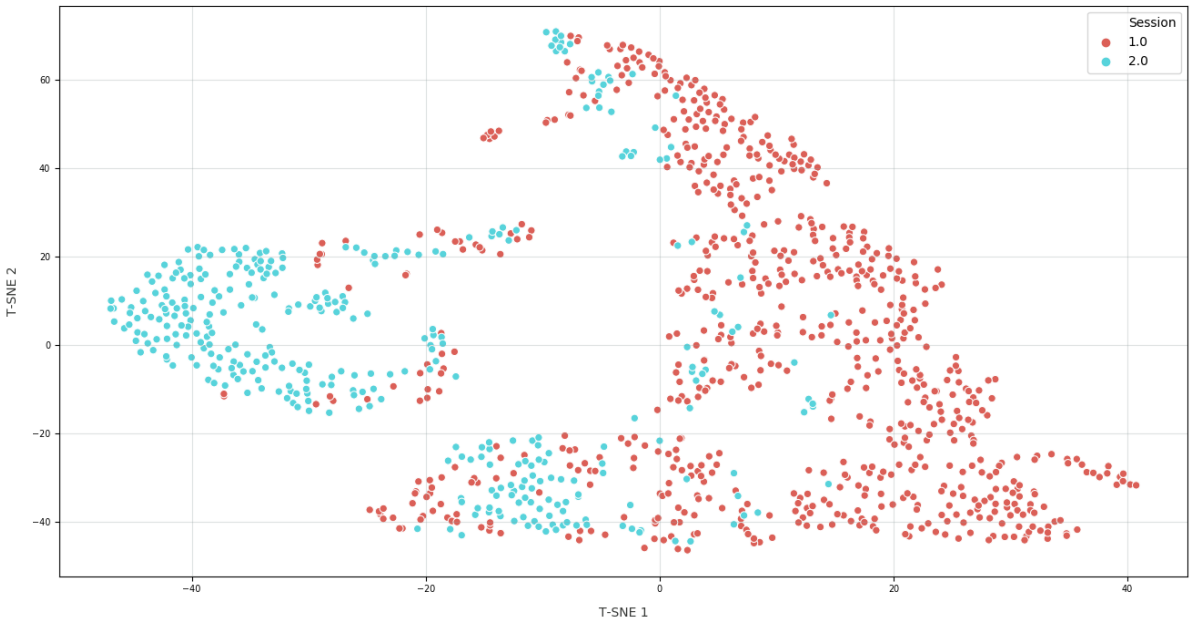


Figure C.13: Visualization of the two-dimensional embedded space that results from the application of t-SNE (Perplexity = 30, Number of iterations = 2000) to the data collected from a single user (user 2 in this case, although the results obtained were similar for all users) during the workload task in both experimental sessions, where each session is represented by a different color.