

Designing Emotionally Expressive Behaviors for an Appearance-Constrained Robot

Evaluating the Affective Interpretation of Motion, Light and Sound

Fernando Corte Vargas



Designing Emotionally Expressive Behaviors for an Appearance-Constrained Robot

Evaluating the Affective Interpretation of Motion, Light and Sound

by

Fernando Corte Vargas

in partial fulfillment of the requirements for the degree of

Master of Science
in Robotics

at the Delft University of Technology,
to be defended publicly on June 28, 2024 at 11:00.

Student number:	4674529	
Chair:	Dr. Jens Kober	Delft University of Technology
Supervisors:	Dr.ir. Joost Broekens	Leiden University
	Bernhard Hilpert	Leiden University
External Committee Member:	Dr.ir. Yke Bauke Eisma	Delft University of Technology

This thesis is confidential and cannot be made public until June 28, 2024

An electronic version of this thesis is available at <http://repository.tudelft.nl/>.



Abstract

How can robots without expressive faces or bodies convey emotions? Why would it be useful if robots could express emotion? In the context of human-robot interaction, could emotional expression lead to a greater comprehension of robotic behaviors and intents? These are questions addressed by the field of affective robotics, which seeks to develop and establish naturalistic social interaction between robots and humans. Emotions can provide a natural communication modality to augment the multi-modal capabilities of social robots in a variety of domains.

Historically, the emphasis in the field has been on facial and bodily expressions, relying heavily on anthropomorphic or zoomorphic robot appearances. This presents a challenge, as most robots are designed with functionality in mind, often lacking expressive faces and bodies, which limits their ability to effectively convey emotions. This study investigates the potential for appearance-constrained robots to convey emotions through variations in motion, light, and sound parameters.

We conducted an experiment where participants rated the emotional qualities of a non-humanoid, faceless robot's behaviors, which were manipulated through variations in motion, light, and sound parameters. Our approach is unique in that it adopts a bottom-up methodology similar to the work of Jack et al. [44] on facial expressions. By systematically varying individual features and observing the resultant emotional perceptions, we aimed to discern the specific affective contributions of each parameter. Using machine-learning based regression models, we sought to predict the perceived emotional qualities based on these systematically varied parameters.

Our findings reveal that variations in motion parameters, particularly speed, significantly influence the perceived intensity of arousal, joy, and dominance. Light temperature was found to affect the perceived intensity of anger and joy, while sound pitch influenced perceptions of surprise and fear. The regression models showed varying degrees of success, with the random forest models often outperforming linear models but also exhibiting a higher tendency to overfit the training data. The linear models, while less prone to overfitting, struggled to capture the full complexity of the emotional responses. These findings suggest that non-anthropomorphic robots can indeed convey emotional qualities through controlled variations in their behaviors, though the strength and clarity of these emotions remain limited. Future research should focus on enhancing the expressiveness of these parameters and testing the models with new data to better understand their generalizability and effectiveness.

Acknowledgments

I would like to express my sincere gratitude to everyone who supported and contributed to the completion of this thesis.

First and foremost, I extend my deepest thanks to my supervisors, Joost and Jens, for their invaluable guidance, insightful feedback, and unwavering support throughout this research journey. Their expertise and encouragement have been instrumental in shaping this work.

I am also profoundly grateful to Bernhard, my daily supervisor, for always being there whenever I needed support. From setting up the experiment to navigating the inevitable bureaucracy of academia to help me graduate on time, your assistance has been indispensable. Your companionship made this journey more enjoyable and less daunting.

A special thanks to the Leiden Institute of Advanced Computer Science at Leiden University for providing the necessary resources and facilities to conduct this research. The collaborative environment, financial support, and access to the necessary equipment were crucial for the successful completion of this study.

I am also thankful to the participants of my study, whose time and insights were invaluable. Their willingness to engage with my research contributed significantly to the findings presented in this thesis.

Last but not least, I am deeply indebted to my family and friends for their unwavering support, patience, and encouragement. To my parents, Martha and Manuel, and my siblings, Sebastián and Paulina, your belief in me has been a constant source of motivation.

To my friends, Andrea, Karla, Armando, Sara, Eilidh, Momo and Mahima, thank you for always being there for me, for all the good times together, and for all the things that are yet to come. Without your support, none of this would have been possible.

And to João, thank you for being my safety net, for always cheering me up when I needed it, and for giving me support and encouragement with loving kindness at the times I did not believe I could do it. I love you from the bottom of my heart.

Thank you all for being part of this journey.

Fernando

Contents

List of Figures	ix
List of Tables	xi
1 Introduction	1
1.1 Background and Motivation	1
1.2 Research Objective and Scope	3
2 Related Work	5
2.1 Theories of Emotion applied to Affective Robotics	5
2.2 Robotic Appearance and Morphology	7
2.3 Modalities of Expression	9
2.4 Emotion Prediction Modeling	13
3 Method	15
3.1 Materials	15
3.2 Base Behaviors	16
3.3 Experimental Variables	19
3.4 Experimental Setup	22
3.5 Analysis	23
4 Results	29
4.1 Data Collection and Processing	29
4.2 Correlation Analysis	29
4.3 Regression Models	30
5 Discussion	41
5.1 Research Questions	41
5.2 Limitations	44
5.3 Recommendations	46
5.4 Conclusion	46
Bibliography	49
A Code	57
B Survey	103
C Human Research Ethics Approval Letter	125
D Risk Assessment	127
E Data Management Plan	141
F Method Details	149
G Results Details	159

List of Figures

1.1	Examples of appearance-constrained robots	2
2.1	Ekman's six basic emotions [30]	6
2.3	The SAM used to measure pleasure (top), arousal and dominance (bottom)	8
2.4	Examples of humanoid robots, sorted by degree of morphological similarity	8
2.5	Examples of appearance-constrained (and therefore non-humanoid) robots	9
2.6	Examples of appearance-constrained robots augmented with light interfaces for emotional expression	11
2.7	Examples of robotic characters from popular motion pictures that make use of NLUs	12
3.1	Different build configuration for the mBot	16
3.2	Logic flow diagram of the wander base behavior	17
3.3	Logic flow diagram of the blink base behavior	18
3.4	Logic flow diagram of the beep base behavior	20
3.5	Video recording setup	22
3.6	Screenshots from the videos	22
3.7	AIC and BIC of the GMM over a range of 2 to 10 clusters	25
4.1	Most important features of the best regression model for joy intensity	32
4.2	Most important features of the best regression model for sadness intensity	33
4.3	Most important features of the best regression model for fear intensity	34
4.4	Most important features of the best regression model for anger intensity	36
4.5	Most important features of the best regression model for surprise intensity	37
4.6	Most important features of the best regression model for pleasure	38
4.7	Most important features of the best regression model for dominance	40
F.1	Distribution of the independent variables	154
F.2	Correlation matrix of the independent variables	154
G.1	Distribution of the PAD dimensions variables based on the aggregated dataset	159
G.2	Distribution of the emotional intensity variables based on the aggregated dataset	160
G.3	Box plots of the emotional intensity variables based on the aggregated dataset	160
G.4	Box plots of the PAD dimensions variables based on the aggregated dataset	160
G.5	Self-correlation matrix of the dependent variables based on the aggregated dataset	161
G.6	Correlation matrix of the independent and the aggregated dependent variables	161
G.7	Scatterplots of the linear model trained with X for joy intensity	162
G.8	Scatterplots of the linear model trained with X_{int} for joy intensity	162
G.9	Scatterplots of the linear model trained with X_{gmm} for joy intensity	162
G.10	Scatterplots of the linear model trained with X_{gmm_int} for joy intensity	163
G.11	Scatterplots of the random forest model trained with X for joy intensity	163
G.12	Scatterplots of the random forest model trained with X_{int} for joy intensity	163
G.13	Scatterplots of the random forest model trained with X_{gmm} for joy intensity	164
G.14	Scatterplots of the random forest model trained with X_{gmm_int} for joy intensity	164
G.15	Scatterplots of the linear model trained with X for sadness intensity	165
G.16	Scatterplots of the linear model trained with X_{int} for sadness intensity	165
G.17	Scatterplots of the linear model trained with X_{gmm} for sadness intensity	165
G.18	Scatterplots of the linear model trained with X_{gmm_int} for sadness intensity	166
G.19	Scatterplots of the random forest model trained with X for sadness intensity	166
G.20	Scatterplots of the random forest model trained with X_{int} for sadness intensity	166
G.21	Scatterplots of the random forest model trained with X_{gmm} for sadness intensity	167
G.22	Scatterplots of the random forest model trained with X_{gmm_int} for sadness intensity	167
G.23	Scatterplots of the linear model trained with X for fear intensity	168

G.24 Scatterplots of the linear model trained with X_{int} for fear intensity	168
G.25 Scatterplots of the linear model trained with X_{gmm} for fear intensity	168
G.26 Scatterplots of the linear model trained with X_{gmm_int} for fear intensity	169
G.27 Scatterplots of the random forest model trained with X for fear intensity	169
G.28 Scatterplots of the random forest model trained with X_{int} for fear intensity	169
G.29 Scatterplots of the random forest model trained with X_{gmm} for fear intensity	170
G.30 Scatterplots of the random forest model trained with X_{gmm_int} for fear intensity	170
G.31 Scatterplots of the linear model trained with X for anger intensity	171
G.32 Scatterplots of the linear model trained with X_{int} for anger intensity	171
G.33 Scatterplots of the linear model trained with X_{gmm} for anger intensity	171
G.34 Scatterplots of the linear model trained with X_{gmm_int} for anger intensity	172
G.35 Scatterplots of the random forest model trained with X for anger intensity	172
G.36 Scatterplots of the random forest model trained with X_{int} for anger intensity	172
G.37 Scatterplots of the random forest model trained with X_{gmm} for anger intensity	173
G.38 Scatterplots of the random forest model trained with X_{gmm_int} for anger intensity	173
G.39 Scatterplots of the linear model trained with X for disgust intensity	174
G.40 Scatterplots of the linear model trained with X_{int} for disgust intensity	174
G.41 Scatterplots of the linear model trained with X_{gmm} for disgust intensity	174
G.42 Scatterplots of the linear model trained with X_{gmm_int} for disgust intensity	175
G.43 Scatterplots of the random forest model trained with X for disgust intensity	175
G.44 Scatterplots of the random forest model trained with X_{int} for disgust intensity	175
G.45 Scatterplots of the random forest model trained with X_{gmm} for disgust intensity	176
G.46 Scatterplots of the random forest model trained with X_{gmm_int} for disgust intensity	176
G.47 Scatterplots of the linear model trained with X for surprise intensity	177
G.48 Scatterplots of the linear model trained with X_{int} for surprise intensity	177
G.49 Scatterplots of the linear model trained with X_{gmm} for surprise intensity	177
G.50 Scatterplots of the linear model trained with X_{gmm_int} for surprise intensity	178
G.51 Scatterplots of the random forest model trained with X for surprise intensity	178
G.52 Scatterplots of the random forest model trained with X_{int} for surprise intensity	178
G.53 Scatterplots of the random forest model trained with X_{gmm} for surprise intensity	179
G.54 Scatterplots of the random forest model trained with X_{gmm_int} for surprise intensity	179
G.55 Scatterplots of the linear model trained with X for pleasure	180
G.56 Scatterplots of the linear model trained with X_{int} for pleasure	180
G.57 Scatterplots of the linear model trained with X_{gmm} for pleasure	180
G.58 Scatterplots of the linear model trained with X_{gmm_int} for pleasure	181
G.59 Scatterplots of the random forest model trained with X for pleasure	181
G.60 Scatterplots of the random forest model trained with X_{int} for pleasure	181
G.61 Scatterplots of the random forest model trained with X_{gmm} for pleasure	182
G.62 Scatterplots of the random forest model trained with X_{gmm_int} for pleasure	182
G.63 Scatterplots of the linear model trained with X for arousal	183
G.64 Scatterplots of the linear model trained with X_{int} for arousal	183
G.65 Scatterplots of the linear model trained with X_{gmm} for arousal	183
G.66 Scatterplots of the linear model trained with X_{gmm_int} for arousal	184
G.67 Scatterplots of the random forest model trained with X for arousal	184
G.68 Scatterplots of the random forest model trained with X_{int} for arousal	184
G.69 Scatterplots of the random forest model trained with X_{gmm} for arousal	185
G.70 Scatterplots of the random forest model trained with X_{gmm_int} for arousal	185
G.71 Scatterplots of the linear model trained with X for dominance	186
G.72 Scatterplots of the linear model trained with X_{int} for dominance	186
G.73 Scatterplots of the linear model trained with X_{gmm} for dominance	186
G.74 Scatterplots of the linear model trained with X_{gmm_int} for dominance	187
G.75 Scatterplots of the random forest model trained with X for dominance	187
G.76 Scatterplots of the random forest model trained with X_{int} for dominance	187
G.77 Scatterplots of the random forest model trained with X_{gmm} for dominance	188
G.78 Scatterplots of the random forest model trained with X_{gmm_int} for dominance	188

List of Tables

2.1	Summary of characteristics of motion in terms of affect attribution	11
2.2	Summary of characteristics of light in terms of affect attribution	12
2.3	Summary of characteristics of sound in terms of affect attribution	13
3.1	Wander input parameters	16
3.2	Blink input parameters	18
3.3	Beep input parameters	19
3.4	Bounds of the independent variables and their motivation	20
3.6	High-level hypotheses	24
3.7	Hyperparameter tuning grid of the random forest regressors	26
4.1	Descriptive statistics of the dependent variables post-aggregation	29
4.2	Summary of significant correlations between the independent and dependent variables	30
4.3	Baseline and optimized models of each micro-hypothesis of H4	31
4.4	Performance scores between the optimized and baseline models for joy intensity	31
4.5	Results of the overfitting ratio tests for the regression models of joy intensity	32
4.6	Performance scores between the optimized and baseline models for sadness intensity	32
4.7	Results of the overfitting ratio tests for the regression models of sadness intensity	33
4.8	Performance scores between the optimized and baseline models for fear intensity	34
4.9	Results of the overfitting ratio tests for the regression models of fear intensity	34
4.10	Performance scores between the optimized and baseline models for anger intensity	35
4.11	Results of the overfitting ratio tests for the regression models of anger intensity	35
4.12	Performance scores between the optimized and baseline models for disgust intensity	35
4.13	Performance scores between the optimized and baseline models for surprise intensity	36
4.14	Results of the overfitting ratio tests for the regression models of surprise intensity	36
4.15	Performance scores between the optimized and baseline models for pleasure	37
4.16	Results of the overfitting ratio tests for the regression models of pleasure	38
4.17	Performance scores between the optimized and baseline models for arousal	38
4.18	Results of the overfitting ratio tests for the regression models of arousal	39
4.19	Most significant features of the best regression model for arousal	39
4.20	Performance scores between the optimized and baseline models for dominance	39
4.21	Results of the overfitting ratio tests for the regression models of dominance	40
F.1	Wander Control Variables	149
F.2	Blink Control Variables	150
F.3	Beep Control Variables	151
F.4	Controlled Variables	153
F.5	Detailed micro-hypotheses for RQ1	155
F.6	Detailed micro-hypotheses for RQ2	155
F.7	Detailed micro-hypotheses for RQ3	156
F.8	Detailed micro-hypotheses of H4	157
G.1	Descriptive statistics of the dependent variables pre-aggregation	159

Introduction

The more ubiquitous and autonomous robots become, the more interpretability becomes important for humans to be able to interact with them effectively [82]. According to Graziani et al. [38], an artificial intelligence (AI) system is interpretable if its working principles and outputs can be translated into human-comprehensible language without compromising the system's validity. In the context of human-robot interaction (HRI), interpretability refers to the extent to which the internal state and decision making processes of a robot are understandable to the user [98]. In HRI, achieving interpretability is crucial as it enhances the natural and fluid interaction between humans and robots, making it desirable for human users.

However, due to the diverse range of tasks and environments in which robots operate, it is impossible to preprogram behaviors that guarantee interpretability in all situations. Therefore, robotic agents should possess the capability to learn, adapt, and change their behaviors autonomously or with the input and feedback of a human to maximize interpretability [50]. This requirement implies that we need robots that not only can understand us, but are also easy to understand. Consequently, there is a need to make robot behavior more interpretable to improve mutual understanding.

To address the gap in mutual understanding between humans and robots, the utilization of affective signals can be considered. Emotional expression, as described by Zych and Gogolla [100] and Trevarthen [94], is a dynamic multimodal behavioral pattern that transcends language and species. Emotional expressions serve as social feedback signals and as a means of empathizing with others [39]. In the context of robotic learning, emotions may provide insights into robot intentions, decision-making, and learning progress. Emotional displays also facilitate feedback and engagement [2], and contribute to building trust and affinity between robots and human teachers [82].

1.1. Background and Motivation

Previous efforts in the development of autonomous intelligent machines have focused primarily on the emulation of human cognitive faculties, including problem solving, logic, learning, sensory perception, language, and various cognitive processes [52, 97]. However, a relatively small sample of this research have explored the potential of emotions [59], despite evidence showing that emotions have a significant influence on high-level cognitive processes [14]. Furthermore, emotions not only make interactions more engaging, but also have a considerable impact on how well individuals understand each other. Effective human communication depends on emotional skills, particularly the ability to perceive and convey emotions [90]. As a result, humans depend on their emotional skills for a multitude of cognitive and behavioral tasks, including but not limited to reasoning, problem solving, social interaction, consciousness, memory, learning, and creativity. Considering the role that emotions play in human cognition, there is compelling motivation to delve into the integration of emotional behavior in robotic agents, especially within the context of HRI.

1.1.1. Affective Computing and Human-Robot Interaction

Affective computing is a multidisciplinary field that investigates how technology can inform our understanding of human emotions, influence interactions between humans and technology, design systems to harness emotions to enhance capabilities and transform human-computer interaction (HCI) through sensing and affective strategies [26]. Within this field, experts agree that emotions are essential to enable socially interactive robots to create natural and fluid human-robot interactions. This point of view is supported by several studies [25, 36, 54, 68, 80].

Emotional expression can therefore be seen as an important aspect to consider in HRI, as it fosters engagement and aligns the goals of humans and robots. When robots exhibit emotions that resonate with human expectations and experiences, it cultivates familiarity, encourages natural interactions, and improves understanding of robot behavior [18]. Furthermore, emotional displays induce empathy and help humans form mental models of the robot's internal state, allowing them to interpret the robot's behavior more effectively in a human-like manner, thus facilitating communication and support [22].

1.1.2. Methods for Implementing Emotions in Robots

The implementation of emotions in robots is a multifaceted process that depends on the robot's ability to both elicit and express emotions. The capacity of a robot to elicit (i.e., to simulate), emotions relies on affective computing methods. Affective computing in robotics draws from an array of methodologies including symbolic knowledge representation, cognitive models, fuzzy models, Markov models, neural networks, and reinforcement learning [77]. Traditional approaches like symbolic and cognitive techniques enable robots to perform diverse tasks, however, they are somewhat restricted in their ability to learn through exploration and feedback, particularly in unstructured tasks.

Machine Learning (ML) approaches, in contrast, focus on adaptive learning and environmental interaction. Reinforcement learning (RL), a subset of ML, is particularly noteworthy as it allows robots to learn via trial and error, constantly interacting with their surroundings to identify and select optimal actions [48]. This learning process is integral for a robot to accurately express appropriate emotions, both in intensity and timing, aligning the elicitation process with the robot's learning mechanism.

After the elicitation process, the next requirement is that the robot should be able to express the elicited emotion in an interpretable manner. In this study, the expression of emotions will be the main focus. For anthropomorphic robots (i.e., robots that resemble humans in form), this is relatively straightforward as they have been the primary type of robots used in studies that explore the design of robotic emotional expressions in the context of HRI. In these studies, non-verbal communication methods such as gestures, facial expressions, and gaze play significant roles [76].

1.1.3. Challenges with Appearance-Constrained Robots

However, the majority of robots currently in use are non-humanoid and appearance-constrained, preventing them from using said non-verbal communication methods. Appearance-constrained robots are non-humanoid robots that lack the expressive faces and bodies necessary for the non-verbal cues commonly used in HRI [21]. Appearance-constrained robots are typically designed with a focus on functionality over form, resulting in a limited ability to utilize facial or somatic expressions for emotion conveyance [87]. A few examples of these robots are shown below:

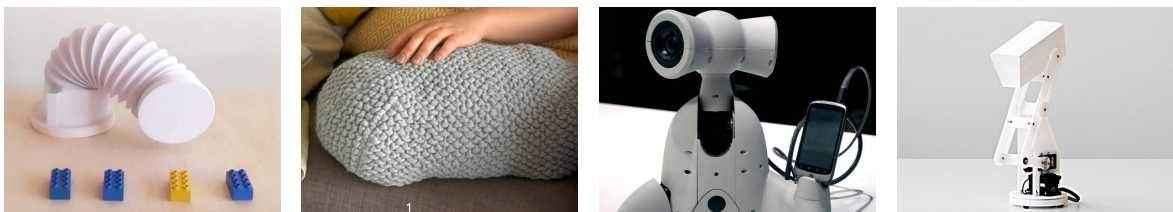


Figure 1.1: Examples of appearance-constrained robots

This limitation presents a significant challenge in HRI. Appearance-constrained robots, despite being the most prevalent variety of robots, are the least capable of interacting with humans in a manner that is perceived as fluid and natural [10]. Their inherent non-anthropomorphic characteristics severely restrict their capacity to convey emotions in a manner comprehensible to human users, which stresses the need for increased exploration of alternative expression modalities for this type of robots.

1.1.4. Alternative Expression Modalities for Appearance-Constrained Robots

Because of the inherent morphological limitations of appearance-constrained robots in conveying emotions through anthropomorphic modalities such as facial expressions, body language, and gait, it is necessary to explore alternative expressive modalities, which include motion, lights, sounds, etc. To determine how effective these modalities can be in improving the ability of appearance-constrained robots to communicate emotions, a variety of design principles and techniques must be explored. The aim is to facilitate the design of expressions that allow appearance-constrained robots to effectively evoke and convey emotions,

thus fostering engaging interactions with human users.

Researchers have investigated various methods through which appearance-constrained robots may be able to communicate emotions by using alternative expressive modalities such as motion, lights, sounds, etc. [85, 86, 87, 75]. The challenge here is not only identifying the specific features of these modalities that could potentially convey emotions, but also in ensuring that they are sufficiently interpretable by human users. Given the inherent variation in how individuals perceive and interpret emotions, and the level of abstraction at which these robots express emotions, it is crucial to assess the effectiveness of different features of these alternative modalities in conveying emotions in a manner that is clear and understandable to humans.

1.2. Research Objective and Scope

Consequently, this study will focus on investigating the relationship between specific parameters within the modalities of motion, light, and sound, and the emotional perceptions attributed to a non-humanoid, faceless robot. Unlike traditional approaches that design expressions or relations upfront and then test the perception, our methodology employs a bottom-up approach. This unbiased method systematically explores how variations in parameters influence emotional perceptions, without preconceived notions about the outcomes.

Based on this, a key aspect of this research is to determine whether it is possible to predict the emotional qualities perceived in the robot's behaviors based on changes in these parameters. By manipulating factors such as speed, light intensity, and sound pitch across the robot's three primary behavior modalities (motion, light, and sound), we aim to establish a quantitative link between these input parameters and the emotional qualities they convey.

To achieve this, we will analyze how changes in each parameter influence the intensity of Ekman's basic emotions and the emotional dimensions of pleasure, arousal, and dominance (PAD). This methodological approach allows for a comprehensive evaluation of emotional responses across both discrete and dimensional models of affect.

The data for this study will be collected from a sample of participants recruited via Prolific, who will watch videos of the robot displaying behaviors characterized by different combinations of motion, light, and sound parameters. Participants will then rate the intensity and type of emotional qualities perceived in each scenario.

The ultimate goal of this research is not only to enhance our understanding of how non-verbal cues in robots can be interpreted emotionally by humans but also to gather insights that can effectively guide the design of emotionally expressive robots. This enhancement will facilitate more natural and intuitive interactions between humans and robots, particularly in situations where emotional expressions could be beneficial.

1.2.1. Research Questions

Following this, the main objective of this study is to address the following main research question (MRQ).

MRQ: Can variations in motion, light, and sound parameters of the behaviors of a non-humanoid, faceless robot influence the perceived emotional qualities of those behaviors?

To comprehensively address this overarching question, we will break it down into specific subquestions focusing on each modality and the combined effects:

RQ1: How do variations in motion parameters (speed, roundness, cycle rate) influence the perceived emotional qualities of the robot's behavior?

RQ2: How do variations in light parameters (light temperature, change in brightness, tempo) influence the perceived emotional qualities of the robot's behavior?

RQ3: How do variations in sound parameters (pitch, intonation, tempo) influence the perceived emotional qualities of the robot's behavior?

RQ4: Can regression models, trained with features derived from the input parameters of the robot's modal behaviors, effectively predict the perceived emotional qualities of those behaviors?

2

Related Work

To enable robots to convey emotions effectively, it is essential to establish a clear definition of emotion. According to the James-Lange theory of emotion, emotion is commonly defined as a combination of psychological aspects, including subjective experiences, expressive behaviors (such as facial expressions, bodily movements, and verbal communication), and physiological responses (such as changes in heart rate and respiration) [19]. Emotions are widely recognized as central to human psychology [65], and several prominent psychological approaches contribute to our understanding of emotions [39].

These theories significantly contribute to our understanding of social psychology, human cognition, emotions, and how we perceive the world. However, it is less clear how these approaches apply to non-human entities, including animals, plants, and artificial autonomous agents like robots, computers, and inanimate objects. The study of emotions in robotic and artificial agents raises important questions, such as what these agents mean as emotional and how best to study their emotions. Answering these questions is vital for advancing our knowledge of emotions in non-human entities and their interactions with humans.

2.1. Theories of Emotion applied to Affective Robotics

Our understanding of emotions is enriched by several influential psychology approaches. One such approach employs categorical models of emotion, aiming to identify fundamental emotions governed by distinct mechanisms, resulting in unique mental states with quantifiable manifestations. Another approach, known as the appraisal model of emotion, contends that emotions or their constituents originate from the evaluation of stimuli in terms of goal congruence, expectation alignment, controllability, and causal attribution. Psychological constructionist viewpoints regard emotions as integral components of an ongoing, continuously evolving construction process, rather than discrete mental states. These theories propose that emotions arise from cognitive and perceptual elements that are intertwined with other mental processes. Furthermore, social constructionist approaches perceive emotions as socially constructed entities, influenced by sociocultural factors and delimited by participant roles and social contexts [39].

These theories make substantial contributions to the field of social psychology and our understanding of human cognition, emotions, and perceptual paradigms. Nonetheless, it is less evident how these frameworks apply to non-human entities, encompassing not only animals, plants, and other living organisms, but also robots, computers, and inanimate objects. The examination of emotions in robotic agents and other autonomous artificial entities poses pivotal questions. What attributes define these entities as emotional, and what are the most effective methodologies for investigating the emotions of these artificial beings? These inquiries are indispensable for advancing our understanding of the emotional terrain inhabited by non-human entities and the consequences of their interactions with humans.

2.1.1. Categorical Theories of Emotion

According to Ekman [29], each fundamental emotion is an indivisible building block of the mind, uniquely caused by dedicated mechanisms, resulting in specific experiences and expressive behaviors. Ekman identifies six primary emotions: joy, sadness, anger, fear, surprise, and disgust, which are believed to be universally and innately recognized. This idea stems from Darwin's seminal work [6], which suggests that facial expressions are a universal language for conveying internal emotional states, transcending cultural boundaries. The universality hypothesis posits that these six core human emotions are universally expressed through identical facial movements in all cultures, enabling universal recognition. In Figure 2.1 an illustration of Ekman's six basic emotions is presented.

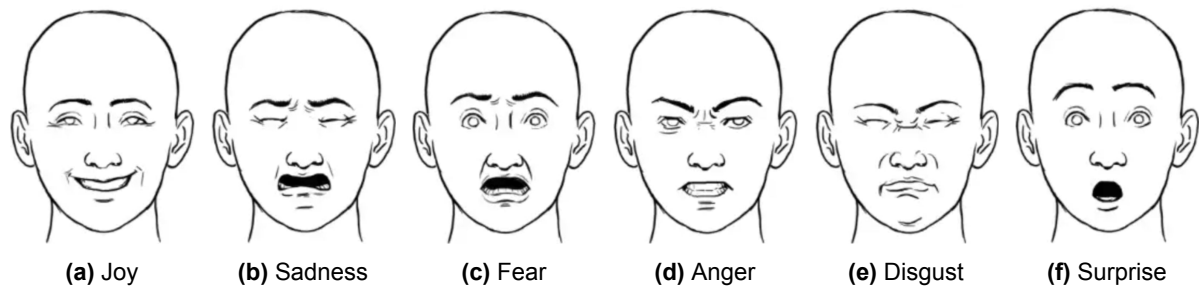


Figure 2.1: Ekman's six basic emotions [30]

Many studies in affective robotics, as noted by Bretan et al. [17], prefer the Ekman model because these emotions are intuitive, easily understood, and conveniently positioned on the core affect plane for regression. Furthermore, Löffler et al. [53] argue that the six basic emotions offer the most potential for intuitive HRI as they are innate, fleeting emotional reactions to external stimuli and are shared between cultures. Some studies adopt Ekman's model simply because it is widely accepted in the scientific community [28].

Other research endeavors aim to explore how a single modality, often motion, can express Ekman's six emotions by varying specific parameters. For example, [92] investigated how shape-changing behaviors involving changes in speed, orientation, fluidity, and movement direction could convey Ekman's basic emotions. Similarly, Schwenk and Arras [78] explored using sound sequences to enhance the gestural display of primary emotions like joy, sadness, and fear, as well as secondary emotions like embarrassment, disappointment, and curiosity.

In general, the choice of categorical emotion models is driven by the assumption that people can intuitively perceive basic emotions without extensive training, minimizing the mental workload [3]. However, many articles challenge these models, particularly because they tend to rely heavily on human facial expressions, raising doubts about their applicability to other modalities [5] and the universal validity of cultural interpretations of facial expressions [44]. Therefore, while studies using these models may yield promising results in emotion recognition for robotic agents, they come with certain limitations that need consideration when evaluating the effectiveness of emotional modeling in robotics.

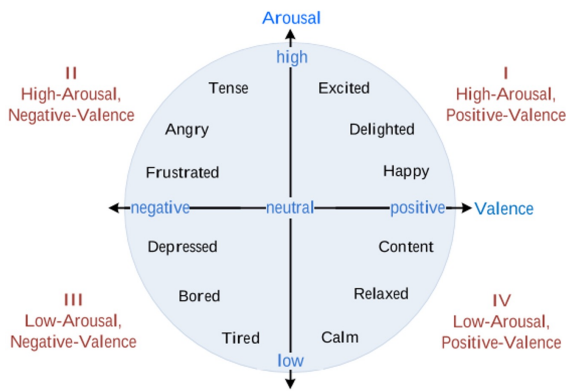
2.1.2. Dimensional Theories of Emotion

In addition to categorical emotion models, some studies focus on using dimensional models. One such model is the circumplex model of affect (Figure 2.2a), developed by Russell in 1980 [72]. Unlike categorical models, the circumplex model does not treat emotions as distinct categories but instead places them on a two-dimensional scale: the arousal (activation) of an emotion and the valence (pleasantness) of an experience. Valence, representing pleasure-displeasure, is depicted horizontally, while arousal, ranging from high arousal to low arousal, is represented vertically. Emotions are evenly distributed around a central point, forming a circular pattern.

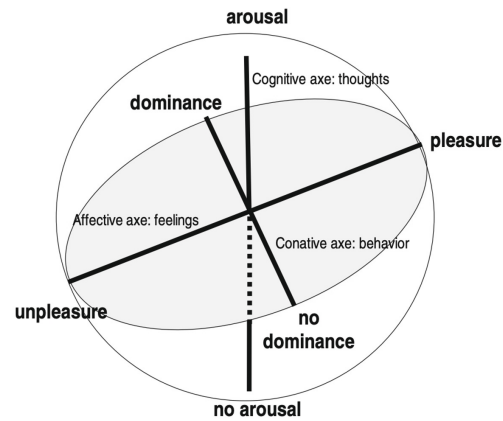
Researchers favoring this model argue that it is valuable for designing emotional expressions and is often seen as a more precise tool for evaluating how accurately emotions are perceived. For example, Strohmeier et al. [89] examined specific shape parameters to study their correlation with valence and arousal in an empirical study. They found that while participants may differ in how they categorize emotions, they tend to agree more when it comes to distinguishing positive vs. negative valence and high vs. low arousal. This suggests that treating emotions as continuous variables within the circumplex model provides a better way to assess emotion perception.

Inspired by this approach, Song and Yamada [85] proposed that emotions within the same quadrant of the valence-arousal space are similar to each other but distinct from emotions in other quadrants. They suggested that focusing on four emotions could best represent the participants' perceptions without introducing unnecessary complexity. Similarly, Feldmaier et al. [33] sought to assess the importance of arousal when portraying simulated sadness or fear compared to joy and anger. Rather than asking participants to categorize emotions, they measured perceived valence and arousal to avoid the bias of affect categorization.

The Pleasure, Arousal and Dominance (PAD) model (Figure 2.2b) is another widely recognized dimensional theory of emotion frequently cited in the literature [57, 73]. According to this model, emotions



(a) Circumplex model of affect [72]



(b) PAD theory of emotion [56]

can be effectively captured within a three-dimensional coordinate system, providing a comprehensive framework. Within the PAD model, general mood types are categorized into eight octants, while emotional states are represented as dynamic 3D vectors. The pleasure component of the PAD model reflects the affective balance, spanning from positive to negative. Arousal signifies the level of physical activity, ranging from excited to calm. Lastly, dominance indicates the degree of control or influence over the environment, ranging from weak to strong [56].

Various studies have embraced the PAD model as a foundational framework to understand and implement emotional theories. For example, Claret et al. [23] and Rincon et al. [69] explicitly reference the PAD model and use it to map motion features such as jerkiness, activity, and gaze to generate emotional movements in robots. Another approach, presented by Nanty and Gelin [60], combines the PAD model with fuzzy logic in a humanoid robot to simulate internal emotional states and achieve emotional coherence over time.

Other studies indirectly employ the PAD model, using measures based on this emotional framework. One such example is the self-assessment manikin (SAM), a non-verbal pictorial assessment technique that directly measures pleasure, arousal, and dominance associated with an individual's affective reaction to diverse stimuli [16]. The works of Bethel and Murphy [12], Klausen et al. [47], Novikova et al. [61], Saerbeck and Bartneck [74], and Tan et al. [92] utilize SAM and draw upon the PAD model in their research.

2.2. Robotic Appearance and Morphology

As this thesis addresses the design of emotional expressions for a robot with appearance constraints, it is imperative to establish clear criteria for classifying different robotic morphologies. A key consideration in this classification is the differentiation between humanoid and non-humanoid robots. A concept that holds significance in this context is "morphological similarity," initially introduced by Epley et al. [32]. Morphological similarity refers to the extent to which observable features of a non-human agent resemble those of a human. Limb configuration constitutes one aspect of this similarity, with humanoid robots featuring limb structures akin to humans, while non-humanoid robots may possess different configurations tailored to their intended functions. Furthermore, humanoid robots often incorporate features that resemble human facial expressions, such as eyes, a mouth, or a realistic face, whereas non-humanoid robots possess unique or specialized features relevant to their specific purposes.

To clarify the boundary between humanoid and non-humanoid characteristics and explore the existing literature on this topic, it is crucial to consider the work of Bethel [9]. Bethel categorizes robot implementations involving affective expressions into three fundamental groups: non-anthropomorphic (or non-humanoid) robots and appearance-constrained robots that employ non-facial and non-verbal affective expression; anthropomorphic (or humanoid) robots that heavily rely on non-facial and non-verbal affective expression; and traditional anthropomorphic robots that combine non-facial and non-verbal affective expressions with conventional facial expressions. An essential term within these categories is "appearance-constrained." As per Bethel and Murphy [11], appearance-constrained robots are not deliberately engineered to be anthropomorphic and lack the ability to exhibit facial expressions or make eye contact. Thus, "appearance-constrained" serves as an additional criterion for distinguishing non-humanoid robots.

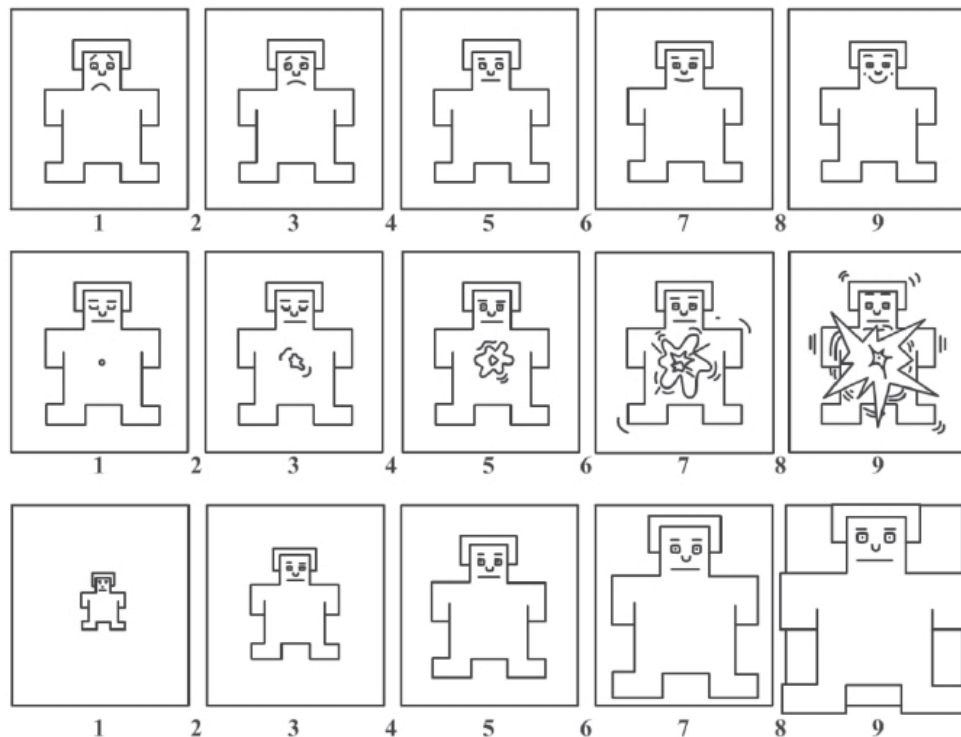


Figure 2.3: The SAM used to measure pleasure (top), arousal and dominance (bottom)

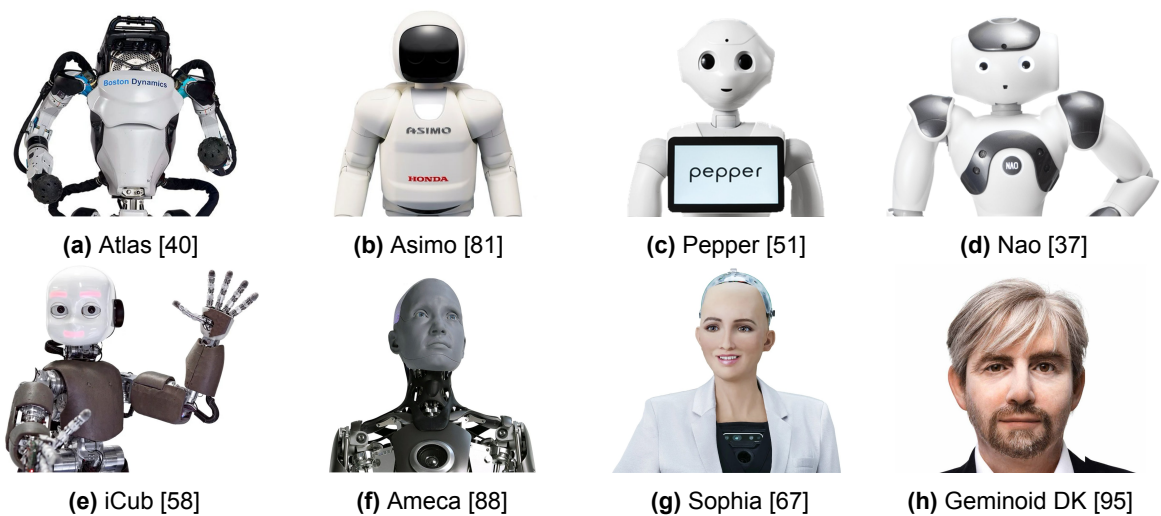


Figure 2.4: Examples of humanoid robots, sorted by degree of morphological similarity

In general, appearance-constrained robots are not engineered to be anthropomorphic or zoomorphic and lack the ability to exhibit facial expressions or make eye contact. Appearance-constrained robots are inherently non-humanoid, non-zoomorphic, and faceless. Consequently, all the robots featured in Figure 2.5 can be regarded as both non-humanoid and appearance-constrained. They are engineered for specific tasks without expressive capabilities in mind, possessing non-human or non-animal morphologies. Due to the inherent morphological limitations of appearance-constrained robots, alternative modalities may be used for the design of emotional expressions. Examples of robots with appearance constraints include devices such as the Greeting Machine [4], robot vacuum cleaners such as the iRobot Roomba [86], Woodie [43], and Post-plant [46], among others.

Concerning the differentiation between humanoid and non-humanoid robots, it is imperative to establish a clear definition for what qualifies as a humanoid robot. According to Siciliano and Khatib [83], humanoid robots draw direct inspiration from human capabilities and deliberately emulate human form and behavior.

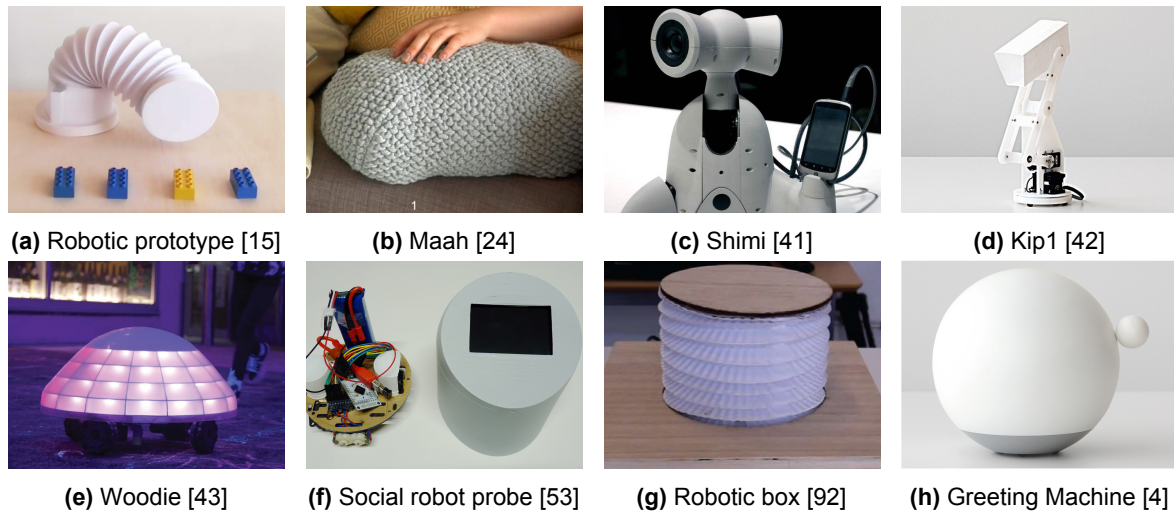


Figure 2.5: Examples of appearance-constrained (and therefore non-humanoid) robots

These robots come in various sizes and shapes, ranging from full-sized legged robots to isolated robotic heads that simulate human sensing and expression. The key emphasis here is on a human-like form, signifying that for a robot to be classified as a humanoid, it must possess a conspicuously human-like shape. Consequently, a high degree of morphological similarity is paramount.

Numerous metrics are employed to gauge anthropomorphism, including assessments of a robot's perceived resemblance to humans. Examples of such metrics include the Godspeed questionnaire [7] and the RoSAS [20], both of which encompass criteria to evaluate anthropomorphism and the perception of human-likeness. Furthermore, Ferrari et al. [34] introduced a method in a preliminary study to classify robots into three groups based on their level of similarity to humans: minimal similarity or non-humanoid (Group 1), moderate similarity or humanoid (Group 2), and high similarity or android (Group 3). As a result, any robot failing to meet the criteria for androids or humanoids is categorized as non-humanoid, including zoomorphic robots.

Based on these definitions, it can be inferred that a non-humanoid robot may possess a body, head, or face, provided that these components significantly diverge from human-like forms. This encompasses a wide spectrum of non-humanoid robots, including zoomorphic robots, aerial robots, underwater robots, search and rescue robots, industrial robots (including robotic arms), autonomous vehicles, mobile robots, and other robotic systems, as long as they distinctly deviate from human-like appearances.

2.3. Modalities of Expression

As discussed in Section 2.2, appearance-constrained robots face limitations in their available output modalities to simulate emotional expression. Depending on the robot, some robots may only have the ability to express emotions through more abstract modalities such as motion, light, sound, haptics, etc. These modalities, as opposed to anthropomorphic modalities, such as facial expressions, body language, and gait, may not immediately be effective in conveying affective signals. Consequently, prior research has focused on establishing the connections between each modality and its ability to effectively communicate specific emotional information. The overarching objective is to assess the viability of using these modalities in the design of appearance-constrained robots and to evaluate their efficiency in conveying emotions.

2.3.1. Motion

Epley et al. [32] highlight the importance of movement as a highly effective modality to achieve anthropomorphism and attributed mental states, including emotional states. However, the ability to communicate emotions through movement is largely based on the degrees of freedom available to the robot performing the action. This is important because appearance-constrained robots can exhibit significant variation in their degrees of freedom, depending on the specific application area and tasks they are designed to perform.

This raises the fundamental question of how individual motion components contribute to different

affective features. According to Kim et al. [46], the speed and size of the movement can effectively convey arousal. Furthermore, they suggest that for valence, if the robot's shape is contorted, shrunken, or twisted and its movement pattern is erratic or unstable, it would be perceived as negative. On the contrary, if all components move smoothly and maintain an open and upright posture, it would be perceived as positive. This concept draws inspiration from plant behavior, where positive emotions are associated with unfurling, standing upright, and opening up, while negative emotions are associated with curling up, rolling inward, bending down, and withering.

Consistent with this study, the research by Tan et al. [91] investigated the kinematic properties related to arousal and valence in emotional states. They found that velocity and range of movement strongly correlate with arousal. Several studies have also indicated that movement orientation serves as a cue for valence. For example, joy can be expressed by moving forward and upward with high speed, while moving backward indicates surprise, shame, or fear. Leaning the body backward is associated with anxiety, panic, and fear, while leaning the whole body forward can represent anger. However, no definitive conclusions have been drawn regarding which kinetic or structural parameters contribute to valence, aside from the orientation of the movement. The authors designed shape-changing movements based on biological motion studies that demonstrated relationships between emotions and shape-changing parameters, such as velocity and orientation.

Furthermore, a study by Löffler et al. [53] devised and validated a set of unimodal and multimodal expressions for basic emotions such as joy, sadness, fear, and anger, using color, motion, and sound as output modalities. They developed metaphorical design guidelines to effectively convey specific emotions through movement. To express joy, metaphors like "joy is up" and "joy is active" guided the design, emphasizing the importance of conveying vitality and positive energy. For sadness, metaphors such as "sadness is down," "sadness is passive," and "sadness is a burden" influenced the design, symbolizing a lack of vitality, passiveness, and reticence. The metaphor "fear is a hidden enemy" translated into "hiding" and "escaping" movements, reflecting the experience of being pursued by fear. Inspired by metaphors for anger such as "anger is hot fluid in a container," "anger is an opponent in a struggle" and "anger is aggressive animal behavior," shaking and forward movements were designed to mimic a bursting container and an inner struggle, as well as fast motion toward the user.

In their analysis, Rooij et al. [71] provide a comprehensive overview of the effects of essential characteristics of affective movement on the attribution of emotions. They highlight several key findings of previous studies. First, they note that perceived instability tends to lead to negative emotion attribution, while perceived stability is associated with positive emotion attribution, as demonstrated by Pavlova et al. [64]. Furthermore, researchers consistently find that acceleration is associated with perceived arousal, which aligns with the findings of Saerbeck and Bartneck [74]. Based on the hypothesis proposed by Kim et al. [46], Rooij et al. [71] suggest that velocity contributes to arousal, smoothness of movement is associated with positive emotions, and size of the movement plays a significant role in both valence and arousal. For example, fast and jerky movements are attributed to joy, whereas large, fast and jerky movements are linked to anger. On the contrary, small and slow movements are associated with sadness and fear, aligning with the metaphors presented by Löffler et al. [53]. Furthermore, round movements tend to evoke positive emotions, while sharp and angular movements are more likely to elicit negative emotions, supporting the hypotheses presented by both Saerbeck and Bartneck [74] and Song and Yamada [86]. Furthermore, Rooij et al. [71] highlight social attributions related to relative motion and perceived goal direction, such as animacy, which contribute to emotional attributions by influencing perceived intentionality.

It is evident that drawing inspiration from biology provides valuable insight for designing robot movements. However, according to Papenmeier [63], animation principles can also be used to create lifelike movements, convey foresight and intelligence, and enhance the expression of intent in robots. One significant advantage of using animation principles is their adaptability to different robot forms, including non-biologically inspired structures like drones. Nevertheless, with a wide range of techniques proposed in the literature and a lack of comparative studies, choosing the appropriate methodology to implement expressive movement can be a daunting task for designers. There is no one-size-fits-all approach, as different strategies are suitable depending on the desired level of expressiveness and the specific robot morphology.

Table 2.1: Summary of characteristics of motion in terms of affect attribution

Modality	Characteristics in Affect Attribution
Motion	<p>Rounded movement patterns convey positive valence.</p> <p>Sharp and angular movement patterns convey negative valence and high arousal.</p> <p>High speed is perceived as more positive with round movement patterns.</p> <p>High speed conveys higher arousal, regardless of movement patterns.</p> <p>Acceleration and curvature also influence valence.</p> <p>Perceived instability lead to negative emotion attribution.</p> <p>Perceived stability is associated with positive emotion attribution.</p> <p>Fast and jerky movements are attributed to joy.</p> <p>Large, fast, and jerky movements are attributed to anger.</p> <p>Small and slow movements are attributed to sadness and fear.</p>

2.3.2. Light

Lights are a versatile and promising visual tool with widespread applications in various machines, such as computers, vehicles, handheld devices, and robots. They are used primarily as simple indicators, offering benefits such as simplicity, readability, and adjustability in terms of visibility [55]. This adaptability enables lights to serve as effective signaling mechanisms for different levels of notification and criticality. Although the core design of most appearance-constrained robots does not incorporate the ability to utilize light and color for emotional displays, it may sometimes be possible to augment them with the necessary interfaces to exploit the potential benefits of light and color, as shown in Figure 2.6.

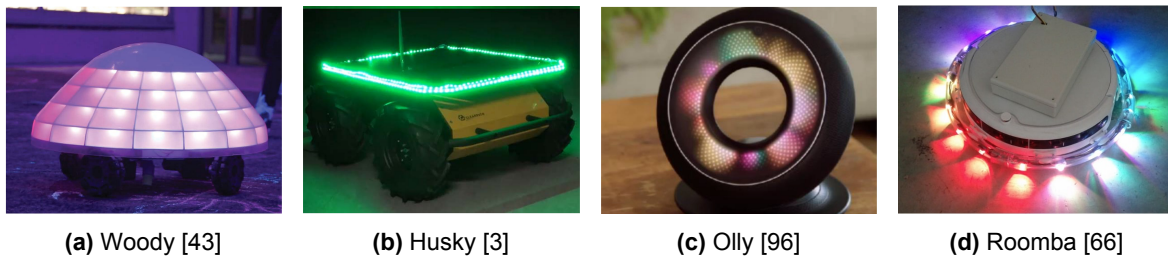


Figure 2.6: Examples of appearance-constrained robots augmented with light interfaces for emotional expression

Betella et al. [8] explored using light parameters such as color, pattern motion, and pattern complexity to express emotions and affective states, based on previous studies that connected colors with emotional attribution [1]. Cool colors (e.g., blue, green) tend to evoke positive and soothing feelings, while warm colors (e.g., yellow, orange) are associated with negativity and physical arousal. The concept of complementary color schemes and the influence of light pattern motion and complexity on valence, interest, and arousal were also examined. These variables were condensed into three abstract motifs (warm, cool, and complementary) and one neutral motif (plain white). The participants then evaluated the emotions attributed to these motifs.

The results of this study showed that color significantly influenced the valence and arousal ratings. Warm colors received lower valence ratings compared to complementary and cool colors, while cool colors received lower arousal ratings compared to warm and complementary colors. The neutral motif produced lower arousal ratings compared to the warm motif. Furthermore, there was a positive correlation between arousal and the motion of light patterns. This suggests that using colors to express affective states can potentially increase the interpretability of emotional states in robots.

Moreover, several studies have explored the importance and effectiveness of light and color in a multimodal context compared to other modalities like sound and vibration. For example, Song and Yamada [85] studied the impact of color, sound, and vibration on human emotional perception of an appearance-constrained robot. They found conflicting mappings between emotional states and colors in the literature, but they selected mappings that received support from the majority of studies. Ultimately,

color emerged as the most important modality for expressing affect, although they noted challenges in expressing joy effectively.

However, Löffler et al. [53] challenged these findings by systematically designing and validating unimodal and multimodal expressions for basic emotions using color, motion, and sound. Surprisingly, unimodal use of color was found to be the weakest predictor of emotion classification, whereas motion and sound performed equally well. Combining color and motion significantly improved emotion classification accuracy and participants' confidence in identifying the robot's emotion. The authors recommended choosing output modalities based on the specific emotions to be conveyed, as no single modality covers all emotions equally well.

Table 2.2: Summary of characteristics of light in terms of affect attribution

Modality	Characteristics in Affect Attribution
Light	Warm colors convey negative valence and high arousal.
	Cool colors convey positive valence and low arousal.
	Positive correlation between arousal and motion of light patterns.
	Multimodal use of light and motion leads to improved accuracy in emotion classification.

2.3.3. Sound

Sound serves as a versatile means of conveying both practical and emotional information in the realm of robot design. Robots naturally generate noise during their operations, but they can also utilize explicit auditory signals, including simple non-linguistic utterances (NLUs) like beeps or chirps, as well as speech. However, while speech is a familiar mode of human communication, it may be inefficient for non-humanoid robots and create unrealistic expectations beyond their expressive capabilities. Therefore, Cha et al. [21] suggests that NLUs could serve as a viable alternative means of expression for non-humanoid robots.

Synthetic NLUs, reminiscent of sounds used by fictional non-humanoid robots, offer a consistent and fitting form of communication. Examples include sound clips from iconic robot movie scenes featuring characters like Wall-E, R2-D2, BB8, and Transformers, which effectively use sound to convey the robot's internal state. Despite its potential, research has shown that the unimodal use of sound may not efficiently convey affective information compared to the use of multiple modalities. This limitation might be attributed to various constraints that render the sound less suitable when used in isolation, as discussed below.

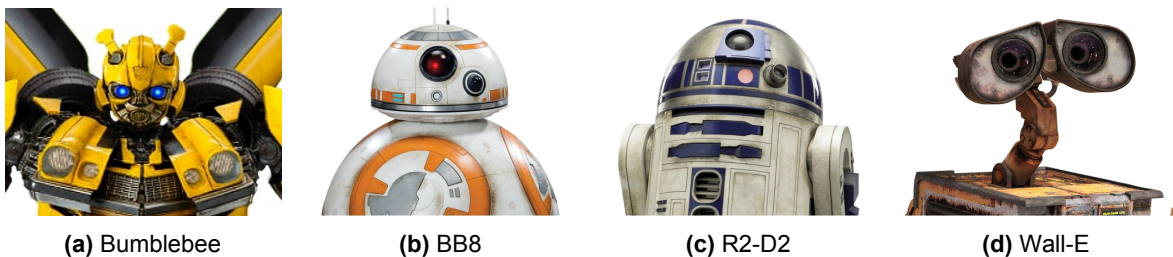


Figure 2.7: Examples of robotic characters from popular motion pictures that make use of NLUs

In their study, Jeong et al. [45] examined how people perceive the functional and emotional intentions conveyed by robot sounds and whether these perceptions are influenced by prior exposure to science fiction movies. The results revealed a significant gap between the intended emotional messages of the sounds and how they were actually interpreted. Participants struggled to correctly identify "positive" sounds as positive, while "neutral" and "negative" sounds were predominantly correctly classified as neutral or negative. Furthermore, some sounds were mistakenly associated with functional messages when their intention was emotional. These findings highlight concerns about the challenges that users may face in accurately interpreting and categorizing unimodal sound expressions. Furthermore, the emotional nature of sounds can be easily confused and mistakenly perceived as conveying a functional message.

However, Song and Yamada [85] propose that affective sounds, particularly NLUs, play an important role in the design of synthetic expressions of emotions. They refer to previous research [49], which

suggests that when beep sounds with rising intonation are emitted from a computer, people perceive the computer as expressing disagreement, regardless of the duration of the beep. On the contrary, sounds with decreasing intonation and longer duration are interpreted as the computer hesitating. These interpretations essentially represent emotions characterized by negative valence and varying levels of arousal.

Song and Yamada [85] argue that the same principles apply when substituting disagreement and hesitation with emotions such as anger and sadness. Based on their findings, the authors make three key recommendations for using sound. First, when expressing anger, a rising sound and highly intense vibration are recommended. Second, when expressing sadness, a falling sound is recommended. Third, whenever possible, they suggest using multiple modalities instead of relying solely on a single modality, aligning with the recommendations of Löffler et al. [53].

In fact, Löffler et al. [53] drew several conclusions about the perceived confidence in the classification of emotions. Experimental results led to the conclusion that sound is not effective in conveying joy and that including sound in a trimodal expression does not improve classification accuracy or perceived confidence. However, the authors found that sound plays an important role in conveying the emotion of sadness. In contrast, sound had a negative impact on confidence scores for fear classification, discouraging its use for that particular emotion. These findings support their initial hypotheses and design recommendations, suggesting that sound can be beneficial but may not universally apply to all modalities or emotions.

Table 2.3: Summary of characteristics of sound in terms of affect attribution

Modality	Characteristics in Affect Attribution
Sound	<p>The use of NLU's might be more suitable than the use of speech for non-humanoid robots.</p> <p>Unimodal use of sound is insufficient for accurate emotion classification.</p> <p>Sounds can be erroneously interpreted as conveying a functional message.</p> <p>Sounds are better at conveying negative emotions.</p> <p>Rising intonation is attributed to anger.</p> <p>Falling intonation is attributed to sadness.</p>

2.4. Emotion Prediction Modeling

Existing research in emotion prediction modeling provides useful insights that can be used to predict human emotional reactions based on observable data. This section delves into relevant literature that has advanced knowledge in this area, focusing on studies that employ innovative methodologies to model emotional qualities using data from human subjects. Notably, while significant progress has been made in understanding and predicting emotions in various contexts, there is a distinct gap in the application of these models to appearance-constrained robots. These robots present unique challenges for emotion prediction modeling because of their inherent restrictions in expressive qualities. By exploring pioneering studies, we present a theoretical foundation for addressing the complexities of emotion prediction in this specific and relatively unexplored domain.

Firstly, we explore the work by Jack et al. [44], which challenges the universality of facial expressions of emotion through a bottom-up modeling approach. This study highlights the cultural nuances in emotional perception and provides a robust framework for understanding how different facial movements are interpreted across cultures.

Next, we examine the research by Dang et al. [27], which addresses the inherent variability and uncertainty in continuously predicting emotional states. By incorporating inter-rater variability into Gaussian Mixture Regression models, this study offers a novel perspective on the dynamic nature of emotion prediction and underscores the importance of accounting for uncertainty in emotion modeling.

2.4.1. Bottom-Up Modeling of Emotional Representations

The study by Jack et al. [44] challenges the universality of facial expressions of emotion through an innovative bottom-up modeling approach. This approach is highly relevant to our thesis as it provides a robust method for capturing the nuances of emotional expression across different cultures, aligning with our goal of understanding how various parameters influence perceived emotions in robots.

The authors used a computer graphics platform to reconstruct mental representations of six basic emotions (happiness, surprise, fear, disgust, anger, and sadness) by analyzing how individuals from Western and Eastern cultures categorized thousands of facial animations. This method involved using generative grammars to randomly generate all possible three-dimensional facial movements, which were then categorized by observers according to the emotions they perceived. The categorization process captured the subsets of facial movements that correlated with the subjective, culture-specific representations of the six basic emotions in individual observers.

The relevance of this approach lies in its ability to model emotional expressions from the ground up, based on individual and cultural perceptions. By leveraging a similar bottom-up methodology, it may be possible to investigate how specific behavioral parameters may influence perceived emotional qualities. Just as the authors of this study used random facial movements to capture cultural nuances in emotion perception, a study can manipulate various modal behavior variations to explore how its emotional qualities are interpreted by human observers.

One of the key aspects of this research that is particularly relevant is the use of a generative grammar framework to explore the entire space of possible facial expressions. This comprehensive approach allowed them to identify specific facial movements that were consistently associated with particular emotions within each cultural context. This systematic variation of behaviors to capture a wide range of possible emotional expressions ensures that findings are grounded in a thorough exploration of the behavioral parameter space. Furthermore, by using the aggregated individual ratings from a sample of human subjects, similar to how the authors aggregated individual facial movement categorizations, a predictive model may be constructed that reflects the collective perception of emotional qualities.

2.4.2. Modeling Uncertainty of Emotion Predictions

Dang et al. [27] address a crucial aspect of emotion prediction systems: the variability and uncertainty inherent in predicting emotional states. Unlike traditional systems that assume constant prediction certainty, this study introduces a novel paradigm to estimate prediction uncertainty using Gaussian Mixture Regression (GMR). This approach leverages inter-rater variability as an indicator of uncertainty, providing a more nuanced understanding of emotional expression over time.

In their study, the authors used the RECOLA database [70], which contains multimodal recordings of spontaneous emotional interactions. Their work is focused on continuous emotion prediction, aiming to understand how emotions fluctuate during interactions. They identified a strong correlation between inter-rater variability (i.e., differences in how different raters perceive and annotate emotions) and prediction uncertainty. By integrating this variability into their GMR model, the researchers were able to more accurately reflect the natural fluctuations in emotional perception. This means that instead of treating all predictions with equal confidence, their model adjusts its certainty based on the observed variability among raters.

Key aspects of their methodology that are particularly relevant to this study include the use of probabilistic models, specifically Gaussian Mixture Models, to capture variability and uncertainty in emotional data. The emphasis on considering inter-rater variability as a measure of uncertainty highlights the role of variability in improving the accuracy and robustness of predictive models. While their study focuses on continuous emotion prediction over time, utilizing the RECOLA database's recordings of emotional interactions, the approach of integrating variability and uncertainty into prediction models is broadly applicable.

Although Dang et al. focused on the dynamic aspect of emotion prediction and the variability over time, their methodology and findings provide a foundational understanding that is relevant to static emotion prediction as well. Their emphasis on probabilistic modeling and the inclusion of uncertainty measures offer valuable insights for improving the accuracy and reliability of emotion prediction systems. This relevance underscores the potential benefits of integrating similar concepts into different contexts of emotion prediction research, such as the static emotion prediction of robotic behaviors explored in this thesis.

3

Method

This chapter outlines the methodology used in this study to investigate how variations in a robot's motion, light, and sound parameters influence perceived emotional qualities. The primary goal of this research is to develop predictive models that can accurately correlate these behavioral parameters with emotional responses, utilizing a bottom-up approach. The study investigates whether changes in motion, light, and sound can significantly alter the perceived emotions conveyed by a non-humanoid, faceless robot. By systematically manipulating and analyzing these parameters, we aim to uncover the underlying relationships that drive emotional perception in human-robot interactions.

To address the research question, we designed an experimental framework that includes details on the robot used, the development of base behaviors for the robot, the measurement of emotional responses, and the training of machine learning models for prediction. The methodology is reported to ensure the clarity, replicability, and validity of our findings, providing a solid foundation for future research and applications in emotionally expressive robots.

3.1. Materials

This section describes the materials used in the study, focusing primarily on the robot used and how it was programmed. This section details the robot's physical characteristics, capabilities, and the rationale for its selection, emphasizing how its design supports the study's objectives.

3.1.1. Robot and Programming

The robot used in this study is the mBot, an educational robot developed by Makeblock, designed to introduce children to the basics of robotics, including mechanical and electronic components, and block-based programming. The mBot kit comprises an mCore main control board (based on Arduino Uno), an ultrasonic sensor, a line-follower sensor, two TT geared motors, an infrared remote controller, and an LED board. The mCore board integrates various sensors (buzzer, light sensor, RGB LED lights, infrared transmitter and receiver), four RJ25 ports, and a USB Type-B port. It supports programming in Arduino C or via the block-based API in mBlock 5, making it ideal for educational purposes.

The mBot in its standard configuration, as illustrated in Figure 3.1a, typically features an ultrasound sensor mounted on its front. This configuration produces a pareidolia effect due to the sensor's resemblance to eyes and the semicircular perforation on the frame mimicking a smile. The taxonomy of robotic appearance in Section 2.2 suggest that the mBot, in its standard configuration, cannot be considered as an appearance-constrained robot. However, various build configurations for the mBot, as depicted in Figure 3.1, offer different levels of appearance constraints. These configurations vary depending on the components attached to the robot's frame.

To ensure that the mBot can be considered as an appearance-constrained robot, specific modifications were implemented. Since the face-like pareidolia primarily stems from the ultrasound sensor and the front perforation, removing or altering these elements can significantly change its appearance. In the case of this study, the robot was not required to avoid obstacles, allowing for the removal of the ultrasound sensor. Additionally, the front perforation was obscured with white duct tape, which further aligned the robot with the concept of an appearance-constrained design.

Furthermore, since light serves as one of the expressive modalities for the mBot in this study, some necessary adjustments were made to the mCore control board. This was necessary because when the

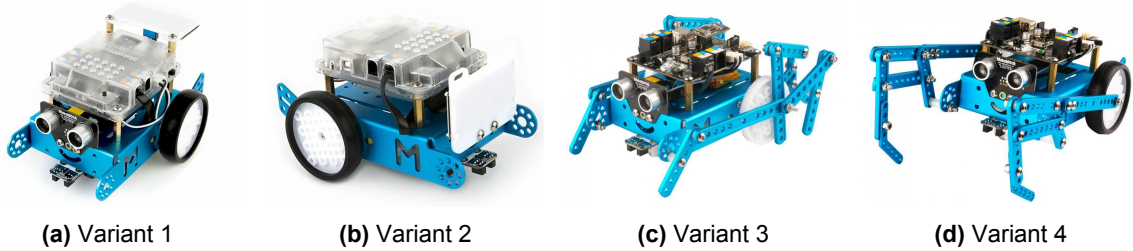


Figure 3.1: Different build configuration for the mBot

mCore is powered on, a red LED light located at the center of the board illuminates and remains active as long as the board is in operation. Moreover, the mCore has a Bluetooth 4.0 dongle that emits intermittent blue flashes to indicate that the control board is not connected to any device via Bluetooth. To ensure that these lights do not introduce biases or affect the interpretation of the light modality generated by the primary RGB LED lights on the control board, the sections of the plastic cover where these lights are located on the mCore control board were concealed with white duct tape.

3.2. Base Behaviors

As outlined in Section 1.2, this study adopts a bottom-up approach to identify whether features in each of the available modalities on the mBot (i.e., motion, light, and sound) can have a significant effect on affect attribution. To do this, three base behaviors for each modality were developed to showcase how different features are expressed in the robot. The selection of these features was guided by comprehensive literature review, as detailed in Tables 2.1, 2.2, and 2.3.

The designed behaviors exhibit a quasi-periodic nature, including input parameters that introduce stochastic patterns in the behaviors to create a natural appearance. The base behavior for motion is named “wander,” metaphorically representing the robot’s action during this behavior. Similarly, the basic behaviors for light and sound are called “blink” and “beep,” respectively. The following subsections will provide a detailed discussion on the design and rationale behind each of these fundamental behaviors, offering insights into their implementation.

3.2.1. Wander

The wander base behavior is a programmed behavior for the robot that allows it to move around in a seemingly random or exploratory manner. The wander behavior is structured into cycles, each consisting of a forward-moving segment and a turning segment. The cycle is repeated for the duration of the behavior.

Input Parameters

This behavior is influenced by several input parameters that define how the robot will move, how long it will move in a certain way, and whether it will stay within the bounds of a predefined area:

Table 3.1: Wander input parameters

Parameter	Description	Valid Range
<code>duration</code>	Total duration (in seconds) for which the wander behavior runs.	$[0, \infty)$
<code>stayInBounds</code>	Boolean flag to keep the robot within an enclosed area with black lines.	$\{\text{True}, \text{False}\}$
<code>wanderSpeed</code>	Base speed of the robot while wandering (in % of motor power).	$[25, 100]$
<code>wanderSlope</code>	Change in speed over the forward-moving segment.	$[-5, 5]$
<code>wanderRoundness</code>	Controls how rounded or sharp the turns are.	$[0, 1]$
<code>wanderCycleRate</code>	Frequency of switching between moving forward and turning (in cycles per second, or Hz).	$(0, \infty)$

Continued on the next page

Table 3.1: Continued from previous page

Parameter	Description	Valid Range
wanderCycleStandard-Deviation	Standard deviation of the variability in cycle duration.	$[0, \infty)$
wanderSpeedStandard-Deviation	Standard deviation of the variability in wander speed.	$[0, \infty)$
wanderPhase	Delay or phase shift before starting the wander behavior (in seconds).	$[0, \infty)$

Behavior Execution

As mentioned before, the wander base behavior combines structured cyclical movements with dynamic adjustments to account for input variability. This results in a robot that moves in an exploratory and somewhat unpredictable path, which matches the metaphorical concept of a wandering robot.

To visualize the execution of the wander base behavior, the logic flow diagram on Figure 3.2 is presented. This diagram focuses on outlining the decision-making process involved in the wander base behavior, such as how to react to environmental boundaries, when to move forward or turn, and how to adapt the behavior over time. A detailed description of the control variables, the logic flow, and the complete code of the wander base behavior can be found in Appendix Sections F.1 and A.1 respectively.

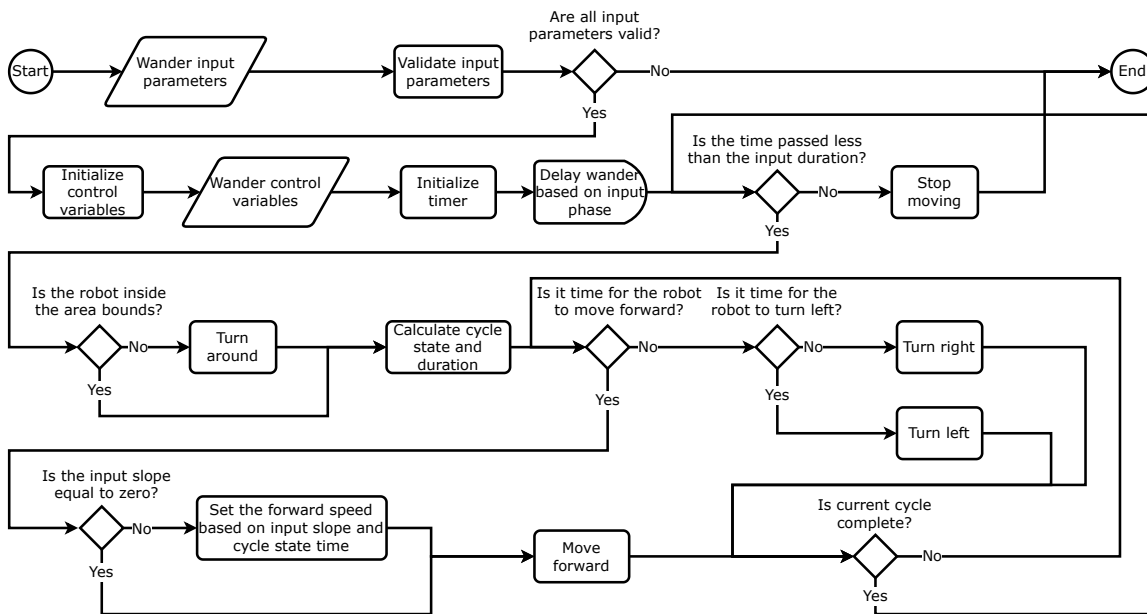


Figure 3.2: Logic flow diagram of the wander base behavior

3.2.2. Blink

Similar to the wander base behavior, the blink base behavior is a programmed behavior that allows the robot to generate repetitive blinking light patterns with various parameters controlling light intensity, timing, and randomness. It can create a wide range of blink patterns based on the input parameters, and is also structured into cycles consisting of a lights-on segment and lights-off segment.

Input Parameters

The blink base behavior is influenced by several input parameters that define how the lighting patterns will look. It can also be noted that most of these input parameters are analogous to the ones of the wander base behavior. This was an intentional design decision that was made to simplify the execution of the blink base behavior, and to keep the code as consistent as possible.

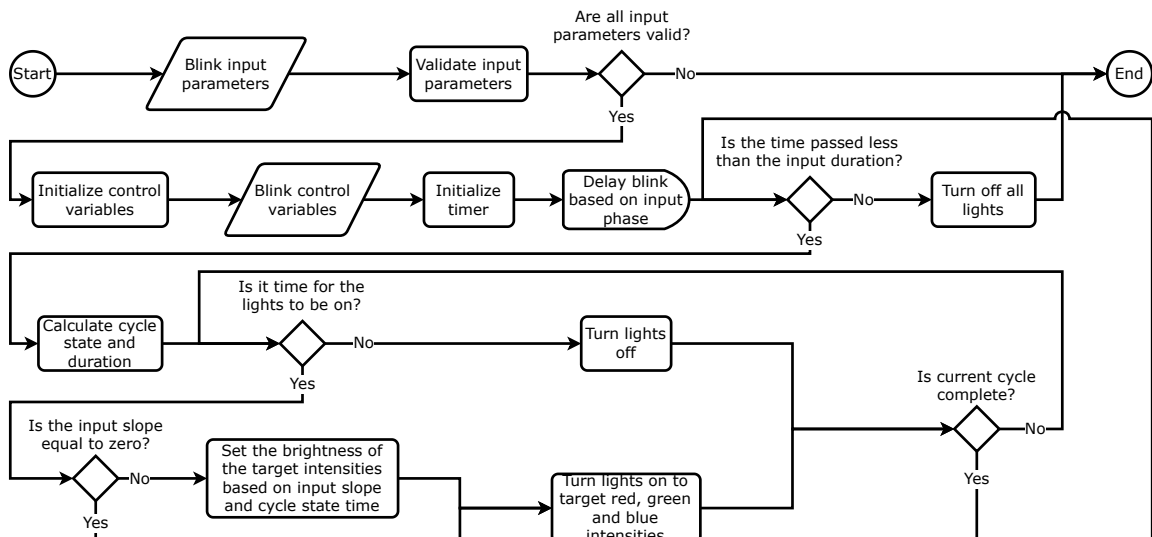
Table 3.2: Blink input parameters

Parameter	Description	Valid Range
duration	Total duration (in seconds) for which the blink behavior runs.	$[0, \infty)$
blinkTemperature	Target temperature of the lights during the blink behavior, controlling the color.	$[0, 1]$
blinkSlope	Determines how the brightness of the lights changes over time.	$[-5, 5]$
blinkLightsOnToOff-Ratio	Ratio of lights-on time to lights-off time in each cycle.	$[0, 1]$
blinkCycleRate	Rate at which the blink cycles occur (in Hz).	$[0, \infty)$
blinkCycleStandard-Deviation	Introduces random variation to the cycle duration.	$[0, \infty)$
blinkTemperature-StandardDeviation	Introduces random variation to the target temperature.	$[0, \infty)$
blinkPhase	Allows delaying the start of the behavior (in seconds).	$[0, \infty)$

Behavior Execution

Similar to the wander base behavior, the blink base behavior consists of an alternating pattern of lights being on and off, controlled by the lights-on-to-off ratio and cycle rate parameters. The brightness during the lights-on segments can vary based on the slope parameter. Randomness can be introduced to add unpredictability, and the behavior can adapt over multiple cycles.

To visualize the logic flow of the blink base behavior, the logic flow diagram on Figure 3.3 is presented. This diagram focuses on outlining the decision-making process involved in the blink base behavior, such as how to alternate between turning the LED lights on and off while adjusting brightness, temperature, and cycle durations based on input parameters and random variations. A detailed description of the control variables, the logic flow, and the complete code of the blink base behavior can be found in Appendix Sections F.2 and A.2 respectively.

**Figure 3.3:** Logic flow diagram of the blink base behavior

3.2.3. Beep

Finally, to generate a base behavior based on the sound modality, the metaphoric concepts associated to NLUs were used to design the beep base behavior. This programmed behavior allows the robot to

generate a repetitive beep sound with various parameters controlling pitch, timing, and randomness. Similar to the wander and blink base behaviors, the beep base behavior is also structured into cycles consisting of a sound segment and a silence segment. The key difference between these base behaviors is that the beep base behavior has the added capacity to either play a random sound or stay silent during the silence segment.

Input Parameters

The beep base behavior takes several input parameters that define how the beeping patterns will sound like. All of these input parameters are analogous to those of the blink base behavior, with the exception of the `beepRandomSoundProbability`. This parameter was added to enable the robot to play a random semitone of the `beepPitch` given the probability `beepRandomSoundProbability`.

Table 3.3: Beep input parameters

Parameter	Description	Valid Range
<code>duration</code>	Total duration (in seconds) for which the beep behavior runs.	$[0, \infty)$
<code>beepPitch</code>	Base pitch (frequency) of the beep sound in Hz.	$[80, 3000]$
<code>beepSlope</code>	Determines how the pitch of the beep sound changes over time.	$[-5, 5]$
<code>beepSoundToSilenceRatio</code>	Ratio of the duration of sound to silence in each cycle.	$[0, 1]$
<code>beepCycleRate</code>	Rate at which the beep cycles occur (in Hz).	$[0, \infty)$
<code>beepCycleStandardDeviation</code>	Introduces random variation to the cycle duration.	$[0, \infty)$
<code>beepPitchStandardDeviation</code>	Introduces random variation to the pitch.	$[0, \infty)$
<code>beepRandomSoundProbability</code>	Probability of playing a random note during the silence duration.	$[0, 1]$
<code>beepPhase</code>	Allows delaying the start of the behavior (in seconds).	$[0, \infty)$

Behavior Execution

As mentioned before, the blink base behavior combines structured cyclical movements with dynamic adjustments to account for input variability. This results in a robot that moves in an exploratory and somewhat unpredictable path, which matches the metaphorical concept of a wandering robot.

To visualize the logic flow of the beep base behavior, the logic flow diagram on Figure 3.4 is presented. This diagram focuses on outlining the steps involved in generating the beeping pattern based on various conditions and parameters. Each step is described in sequence, indicating the decision points and actions taken during the execution of the behavior. A detailed description of the control variables, the logic flow, and the complete code of the beep base behavior can be found in Appendix Sections F.3 and A.3 respectively.

3.3. Experimental Variables

In this section, we will introduce and discuss the selection of the controlled, independent, and dependent variables. Additionally, we will detail the methodological considerations and steps taken to ensure the robustness and reliability of the experimental design, including the sampling methods used and the rationale behind the chosen parameters.

3.3.1. Controlled Variables

The controlled variables in this study are a subset of the input parameters of the base behaviors that were set to a constant value to ensure consistency and isolate the effects of the independent variables.

Parameters like `wanderCycleStandardDeviation` are fixed to control the randomness of the base behaviors, ensuring that the base behaviors appear natural. We also set the phase for all behaviors to

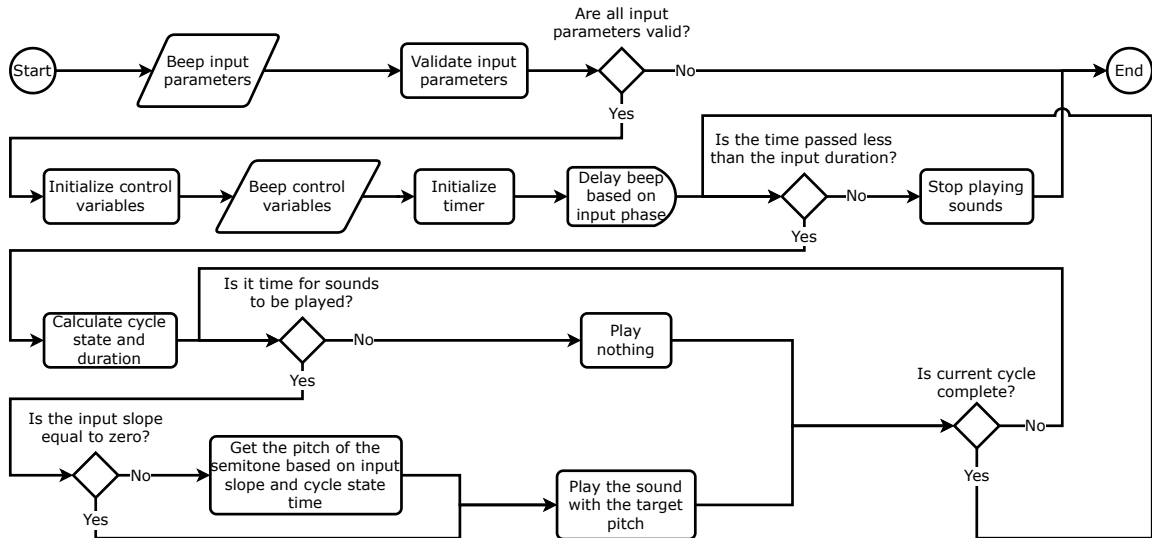


Figure 3.4: Logic flow diagram of the beep base behavior

zero, and `stayInBounds` is set to true. A complete breakdown of the controlled variables and the values they were set to is provided in Appendix Section F.4.

3.3.2. Independent Variables

In this thesis, the independent variables consist of selected input parameters that define the robot's base behaviors as explained in Section 3.2. After careful examination, nine parameters were identified as having the most significant influence on the variability of these behaviors:

- **Wander:** `wanderSpeed`, `wanderRoundness`, `wanderCycleRate`
- **Blink:** `blinkTemperature`, `blinkSlope`, `blinkCycleRate`
- **Beep:** `beepPitch`, `beepSlope`, `beepCycleRate`

These parameters were identified through observational assessments as having the most significant impact on the variability of the robot's behaviors. The decision to limit the study to these nine parameters was driven by a need to balance the comprehensiveness of behavioral variability with practical constraints on sampling complexity. A larger set of parameters would increase the variability but also require a greater number of samples, complicating the experimental design and data analysis.

Since we are looking to manipulate these independent variables, careful consideration must be given to the selection of values for each variable in the experiment. For this, generating samples of these variables that ensure each sample point is spaced to minimize correlation is crucial, as this would maintain the independence among variables.

Moreover, each of these variables is bounded to maximize the variability of the base behaviors without reaching extremes that could adversely affect the behaviors. By setting these bounds, we ensured that the robot's behaviors remained within a range that is both functional and perceivable by human observers. The bounds were selected based on practical observations and constraints:

Table 3.4: Bounds of the independent variables and their motivation

Parameter	Bounds	Motivation
<code>wanderSpeed</code>	[30%, 100%]	Speeds below 30% could result in insufficient motor power, causing the robot to barely move, which is undesirable for the wander behavior.

Continued on the next page

Table 3.4: *Continued from previous page*

Parameter	Bounds	Motivation
beepPitch	[100 Hz, 1000 Hz]	Frequencies below 100 Hz were too deep for the robot to produce, while frequencies above 1000 Hz were too high-pitched and unpleasant for human listeners.
wanderRoundness, blinkTemperature	[0, 1]	These values are inherently between 0 and 1, as explained in Section 3.2.
blinkSlope, beepSlope	{-1, 0, 1}	Continuous values led to unexpected intonations or changes in brightness, and values beyond this range had similar disruptive effects.
wanderCycleRate, blinkCycleRate, beepCycleRate	[0.5 Hz, 6 Hz]	Rates below 0.5 Hz resulted in periods longer than the desired quasi-periodic effect (e.g., a period of 2 seconds is too long for a 20-second video). Rates above 6 Hz resulted in periods shorter than 0.16 seconds, making the behavior too rapid to be perceived as quasi-periodic.

3.3.3. Sampling Method

The sampling of the independent variables was done using Sobol sequences [84]. This method helps ensure that each sample point is spaced to minimize correlation, thereby maintaining the independence among variables. This property is crucial when examining the impact of each independent variable without overlap or undue influence from others.

More details about the implementation of the sampling method and additional proof of non-multicollinearity of the independent variables can be found in Appendix Section F.5. The samples were generated using the Quasi-Montecarlo sub-module of Python's `scipy` library. The complete code can be found in Appendix Section A.4.

3.3.4. Dependent Variables

The dependent variables were selected to measure the emotional qualities of the robot's behaviors based on the perception of a group of human observers. These variables measure emotional perception in three distinct ways: the intensity of categorical emotions, the PAD dimensions, and an open-ended appraisal of the robot's emotional state.

Participants rated the intensity of happiness, sadness, anger, fear, surprise, and disgust as expressed by the robot's behavior. Additionally, participants could select another emotion and rate its intensity if they perceived an emotion not covered by Ekman's six basic categories. The intensity was measured using a 5-point Likert scale: Very Low, Low, Average, High, and Very High, with an option to select N/A if the emotion was not perceived at all. The selection of Ekman's basic Emotions was motivated by past work in the field of affective robotics (as described in Section 2.1.1).

Moreover, the participants rated the PAD dimensions using the SAM (Figure 2.3 [16]). The PAD model is widely recognized for its effectiveness in capturing the core dimensions of emotional experience (see Section 2.1.2). The SAM is particularly suitable for this study because it bypasses language barriers and reduces the cognitive load on participants, making it easier for them to more accurately measure their perception of the emotional qualities of the robot.

Since the emotional intensities are measured on Likert scales and the PAD dimensions are measured using the SAM, we can treat all the dependent variables as interval variables. For the Likert scale data, any N/A responses were treated as zero. This allows for more flexibility in the analysis of this thesis, which ultimately centers on constructing a bottom-up model by manipulating the independent variables and measuring participants' emotional perceptions of the robot using these dependent variables.

Finally, the participants provided open-ended responses about their thoughts or analysis regarding the causes of the robot's emotional reactions in the video, especially how the situation might affect the robot's goals, desires, or overall well-being. Although this was measured in the survey, it is not included in the scope of this thesis. While appraisals offer valuable qualitative insights into participants' interpretations of the robot's behavior, the focus of this thesis is on quantitative analysis of emotional perception.

3.4. Experimental Setup

In this section, we describe the experimental setup designed to investigate how the features of the modalities of light, sound, and motion can lead to the perception of emotions in a robot's behavior. This experiment involved human participants who observed the robot's behaviors and provided feedback on their emotional perceptions. We detail the sampling method, recording setup, data collection process, and participant recruitment strategy.

3.4.1. Recording Setup

We recorded 512 videos, each corresponding to a unique sample of the independent variables. The robot's behaviors were observed and recorded from both a top view and a side view. The robot moved freely within an area bounded by black lines. The videos were recorded in 1080p resolution at 60 frames per second (fps), and each video was 20 seconds long. To ensure high-quality recordings, two tripods were used to ensure that the cameras would stay still and to guarantee the consistency across videos. Figure 3.5 shows a picture of the recording setup, and Figure 3.6 show screenshots from videos as seen from the top and side views.



Figure 3.5: Video recording setup



Figure 3.6: Screenshots from the videos

3.4.2. Data Collection

The data collection was conducted through an online survey using Qualtrics. The survey included an opening statement that provided participants with important information about their rights, the task description, and the expected duration of the survey, which was estimated to be 25 minutes based on a pilot study. A copy of the survey is shown in Appendix B.

Following the opening statement, participants were directed to an informed consent form to ensure they understood the information provided and agreed to participate. The informed consent form included several yes/no questions, confirming that participants had read the opening statement, understood their rights, and agreed to participate under the stated conditions.

To ensure participant engagement and data quality, we included a commitment request in the survey. This request asked participants to commit to spending enough time to provide thoughtful answers. Research by Qualtrics indicates that commitment requests are more effective than attention check questions for ensuring participant engagement [35].

Participants then answered two demographic questions (country of origin and age) and were presented with an example block of questions to familiarize them with the survey format. This example block included snippets of example robot behaviors and detailed explanations of the survey questions. The survey questions were design to measure the perception of the participants on the emotional qualities of the robot's behavior (as described in Section 3.3.4).

The instructions in the survey were carefully written to ensure that the language was clear, and that the emphasis lied on what emotional quality the robot seemed to express through its behavior. The main goal was to avoid that the participants would be inclined to believe that the robot was actually feeling any emotions. Instead, the participants were directed to share their interpretation of the emotional qualities of the robot's behavior.

Each participant rated a random subset of 10 videos out of the 512 recorded. Qualtrics ensured that each video was rated uniformly across participants. This randomization process helped in gathering a representative set of ratings for each video.

3.4.3. Participant Recruitment

Participants were recruited through Prolific, an online platform known for its high-quality participant pool and transparent compensation practices. Initially, Amazon MTurk was considered, but Prolific was chosen due to better customer service and higher data quality. The recruitment goal was 312 participants, calculated to ensure at least six ratings per video. Each participant rated 10 videos, resulting in sufficient data for each video.

To ensure data quality and to limit the demographics of the study to a Western focus, some screening filters were applied. As such, the participants had to be 18 years or older. Additionally, they had to have at least a 98% approval rate on Prolific, with a minimum of 100 completed tasks. Finally, participants were also required to be fluent in English, further supported by residence in a country where English is either the official language or has a very high rate of fluency: Ireland, Australia, Canada, Denmark, Finland, the Netherlands, New Zealand, Norway, Singapore, Sweden, the UK, or the US. Participants received £3.75 GBP for their participation, based on Prolific's suggested hourly rate of £9.00 GBP and the median completion time of 25 minutes.

3.4.4. Ethics Statement

This study was conducted with the approval of the Human Research Ethics Committee (HREC) of the TU Delft. The letter of approval can be found in Appendix C. Prior to data collection, a comprehensive risk analysis and a data management plan were developed to ensure the ethical handling and protection of participant data. These documents detail the measures taken to mitigate potential risks and ensure data integrity, confidentiality, and compliance with relevant ethical standards. The full risk analysis and data management plan are included in Appendices D and E respectively.

3.5. Analysis

To effectively address the research questions outlined in Section 1.2.1, a clear specification of the analysis methodology is necessary. This section details the statistical methods used for statistical analysis, hypothesis testing, and modeling. The code used for the analysis can be found in Section A.5.

3.5.1. Hypotheses

In this subsection, we formulate specific hypotheses to help us answer the research questions outlined in Section 1.2.1:

- RQ1:** How do variations in motion parameters (speed, roundness, cycle rate) influence the perceived emotional qualities of the robot's behavior?
- RQ2:** How do variations in light parameters (light temperature, change in brightness, tempo) influence the perceived emotional qualities of the robot's behavior?
- RQ3:** How do variations in sound parameters (pitch, intonation, tempo) influence the perceived emotional qualities of the robot's behavior?

- RQ4:** Can regression models, trained with features derived from the input parameters of the robot’s modal behaviors, effectively predict the perceived emotional qualities of those behaviors?

To do this, we define hypotheses that can test the relationships between the parameters of the robot’s base behavior and the participants’ emotional perception. Each hypothesis is directional and tied directly to the independent and dependent variables. Since we are looking to test for directional associations, we can formulate each hypothesis in statistical terms as null and alternative hypotheses. This approach ensures that the study’s outcomes can directly inform the effectiveness and impact of specific robotic behaviors on perceived emotions.

The definition of these hypotheses is intentionally broad to provide a high-level overview of the questions we aim to address. To effectively test these hypotheses, we have defined more detailed micro-hypotheses, which are outlined in Appendix Section F.6. The results of testing these micro-hypotheses will inform and support our conclusions regarding the broader, high-level hypotheses.

Table 3.6: High-level hypotheses

ID	Hypothesis
H1	Variations in motion, parameterized by speed, roundness, and cycle rate, significantly influence the perception of the emotional qualities of the robot’s behavior, measured by the intensities of categorical emotions and PAD dimensions.
H2	Variations in light, parameterized by light temperature, change in brightness, and tempo, significantly influence the perception of the emotional qualities of the robot’s behavior, measured by the intensities of categorical emotions and PAD dimensions.
H3	Variations in sound, parameterized by pitch, intonation, and tempo, significantly influence the perception of the emotional qualities of the robot’s behavior, measured by the intensities of categorical emotions and PAD dimensions.
H4	Regression models can effectively predict the perceived emotional qualities, represented by the intensities of categorical emotions and PAD dimensions, using features derived from the input parameters of the robot’s modal behaviors.

3.5.2. Data Preparation

To ensure the integrity and quality of the dataset, a comprehensive data cleaning and preprocessing phase was necessary. This involved identifying missing values, and detecting and removing outliers using the interquartile range (IQR) method. This was done because outliers can significantly skew results and affect the performance of statistical models.

As was mentioned earlier, we treated all dependent variables as interval data. This means that the values of the dependent variables that were measured in Likert scales were encoded using numeric values from 1 to 5. Since the participants also had the option to select “N/A,” these values were encoded as zero. This was considered a valid imputation method, because the instruction for the participant was to select N/A if they did not see a specific emotion in that video. This practically means that the intensity of said emotion was equivalent to zero.

3.5.3. Noise Reduction

One of the most significant challenges in the analysis was the inherent noisiness of the dependent variables. Given that the dependent variables measure the perceived emotional qualities of an inanimate object, it is not a surprise that these variables are noisy. This noise essentially encodes the variability in personal opinions and subjectivity inherent to the perception of emotions.

To address this, we decided to group the participants’ ratings by video, and calculate the mean for each dependent variable across all 512 videos. Aggregating these ratings helped reduce variability and highlight consistent patterns in the data. This approach aimed to simplify the dependent variables and mitigate the impact of noise, leading to potentially more accurate predictions.

Although this step reduced the number of data points to 512 (since we only have 512 videos), it was done to ensure a clearer signal for the models to learn from. Aggregating the data in this manner was done with the hope of revealing the underlying trends and relationships between the robot's behavioral parameters and the perceived emotional qualities, enhancing the robustness of the analysis.

3.5.4. Model Selection

In order to construct a predictive model of the emotional qualities based on the robot's behavioral parameters, we selected two types of regression models. The selection of these models was guided by their ability to capture different types of relationships within the data, ranging from simple linear associations to complex non-linear interactions.

Linear regression was chosen as due to its simplicity and interpretability. It assumes a linear relationship between the independent and dependent variables, making it a straightforward approach to understand the direct effects of each parameter. Linear regression provides clear insights into how each independent variable contributes to the prediction of the dependent variables, offering a transparent model that can be easily interpreted and communicated. However, while the model is simple and interpretable, it may not capture complex non-linear relationships effectively.

The random forest regressor was selected for its ability to handle non-linear relationships and interactions between variables. As an ensemble learning method, random forests builds multiple decision trees and merges them to improve prediction accuracy and control overfitting. This model is particularly robust to noise and outliers, making it suitable for our dataset which may contain complex patterns that are not easily captured by linear models. Additionally, random forests can provide insights into variable importance, helping to identify which features are most influential in predicting the emotional qualities.

3.5.5. Feature Engineering

To mitigate the risk that the models would not be able to accurately predict the emotional qualities of the robot's modal behaviors based on the wander, blink, and beep parameters alone, we considered extensive feature engineering. These features were designed to enhance the models' ability to capture non-linear relationships and interactions between variables. The training dataset that includes these features is later referred to as x .

Firstly, we derived interaction terms between the nine input parameters of the three modal base behaviors. These interaction terms, calculated as the product of each input parameter with every other input parameter (excluding itself), were designed to capture more complex interactions between the input parameters. The training dataset that includes these features is later referred to as X_{int} .

Subsequently, we used a Gaussian Mixture Model (GMM) on the independent variables to calculate cluster membership probabilities to encode the underlying structures and patterns in the data. This approach is inspired by studies that utilize unsupervised learning techniques for feature engineering in training machine learning models [27, 93, 99]. By incorporating these cluster membership probabilities as features, we want to investigate if it can enhance the predictive accuracy of the regression models. The optimal number of clusters was determined by minimizing the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) over a range of 2 to 10 clusters. As can be seen in Figure 3.7, the optimal number of clusters is $n = 6$. The training dataset that includes these features is later referred to as X_{gmm} .

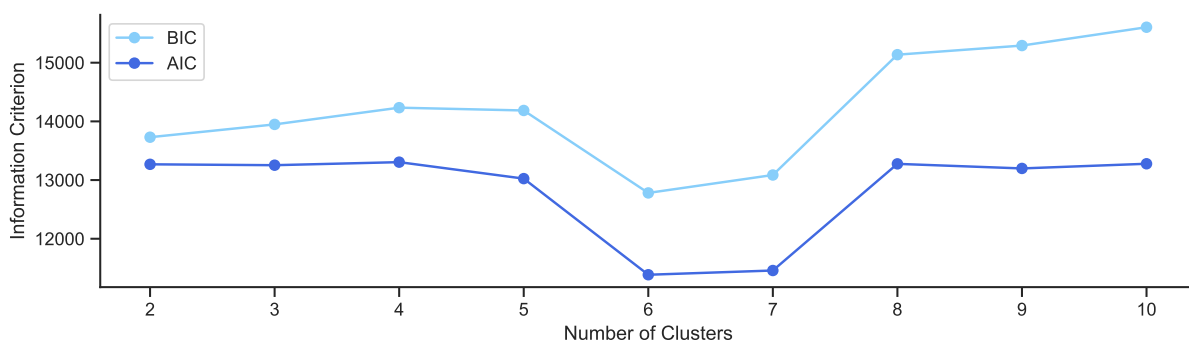


Figure 3.7: AIC and BIC of the GMM over a range of 2 to 10 clusters

Finally, we included interaction terms between the original independent variables and the new GMM cluster membership probabilities. By considering these interactions, we aimed to capture even more complex patterns in the data, ultimately improving the models' ability to generalize and make accurate predictions. The training dataset that includes these features is later referred to as `X_gmm_int`.

3.5.6. Training Approach

For the linear regression model, we initially planned to use forward selection as the stepwise regression method to identify the most relevant independent variables. Forward selection is a method that starts with no variables in the model, adds variables one by one based on a chosen criterion (in our case, the p -values and F -statistics of the univariate linear regression tests), and stops when no additional variables significantly improve the model. This process ensures that the linear model is both parsimonious and effective, capturing the essential relationships between the independent variables and the dependent variables without including unnecessary predictors.

However, in the course of our analysis, we opted for backward selection instead of forward selection, because the former proved to be more efficient in our case, converging much faster than the latter. Unlike forward selection, which starts with an empty model and adds variables, backward selection begins with all variables in the model and removes them one by one based on the p -values and the chosen significance level. This approach also aims to construct a parsimonious model while ensuring that only significant predictors remain.

To ensure robust evaluation of the linear regression models, we used k -fold cross-validation. This technique involves dividing the dataset into k subsets (folds) and training the model k times, each time using a different subset as the validation set and the remaining data as the training set. This process provides a comprehensive assessment of the model's performance across different splits of the data, mitigating the risk of overfitting to a single train-test split. The number of folds was set to $k = 5$ for all regression models.

Since random forests inherently perform feature selection through the construction of multiple decision trees, stepwise regression is not necessary for these models. Instead, we conducted hyperparameter tuning to optimize the performance of the random forest regressors. Table 3.7 provides a summary and motivation of the selected hyperparameter tuning grid. By exploring these different hyperparameters, the hyperparameter tuning grid is designed to find the best combination of parameters that balances model complexity and generalization, thereby helping to train models that do not overfit the data.

Table 3.7: Hyperparameter tuning grid of the random forest regressors

Hyperparameter	Chosen Values	Motivation
Number of trees in the forest	100, 200, 500 trees	Ensures the model's robustness by averaging predictions over many trees.
Maximum depth of each tree	10 to 30 or unlimited	Controls the complexity of the trees and helps prevent overfitting by not allowing trees to grow too deep.
Criteria for splitting nodes	2, 5, 10, 20 samples	Affects how sensitive the model is to noise and variability in the data.
Minimum number of samples required to be at a leaf node	1, 2, 4, 10 samples	Helps avoid overfitting by ensuring that each leaf node has a sufficient number of samples.
Strategy for selecting the number of features to consider when looking for the best split	All features, square root of the number of features, logarithm of the total number of features	Helps to prevent any single feature from having too much influence on the model and introduces randomness to make the model more robust.

We also applied k -fold cross-validation for the random forest models, following the same procedure

as with the linear regression models. This technique ensures a comprehensive evaluation of the model's performance and mitigates the risk of overfitting.

Finally, since we have four training datasets as specified in Section 3.5.5 (X , X_{int} , X_{gmm} , and $X_{gmm_{int}}$), we train each linear and random forest model using these four datasets for each of the nine dependent variables. This means that in the end we will have 36 linear models and 36 random forest models.

3.5.7. Model Evaluation

To be able to provide answers for the hypotheses of RQ4 specified in Section 3.5.1, a clear approach for model evaluation is necessary. We evaluate the performance of our models using the MSE and the R^2 metrics [31]. MSE measures the average squared difference between observed and predicted values, providing an indication of the model's accuracy. A lower MSE indicates better model performance. R^2 indicates the proportion of variance in the dependent variable explained by the independent variables. A higher R^2 value indicates a better fit of the model.

As mentioned above, we use k -fold cross-validation to evaluate model performance. Specifically, we look at the average MSE and R^2 across k -folds to get an approximation of the average performance of the model. Additionally, we select the best model based on the model from those k -folds which minimizes the MSE in the training dataset of all k -splits. This approach ensures that we choose the model with the best performance while mitigating the risk of overfitting to a single train-test split.

Finally, by analyzing both train and test performance, we assess whether the models are overfitting or underfitting the data. Overfitting occurs when a model performs well on the training data but poorly on the test data, indicating that it has learned noise rather than the underlying pattern. Underfitting, on the other hand, occurs when a model performs poorly on both training and test data, indicating that it has not learned the underlying patterns adequately.

3.5.8. Statistical Testing

To test the micro-hypotheses of H1, H2, and H3, a method for correlation analysis must be defined. Since all dependent variables can be considered as interval variables (as explained in Section 3.3), we could consider using Pearson's correlation coefficient. However, Pearson correlation assumes that the variables are normally distributed and is sensitive to outliers. If this assumption is not met, the risk of inflating Type I error rates and reducing power increases [13].

Because we cannot be certain that this assumption is met, the Spearman's rank correlation coefficient with associated p -value is considered more suitable for testing these hypotheses. Spearman's correlation test is a non-parametric test and does not assume any specific distribution of the variables. It is robust to outliers and can be used with ordinal or ranked data. It evaluates if there is a (directional) monotonic relationship between independent variables and dependent variables.

Since we are executing multiple statistical tests, we apply the Bonferroni-correction to the p -values [79]. This is done to apply a correction to prevent inflating the family-wise error rate. This correction helps to mitigate the increased risk of Type I errors (false positives) that arises when multiple statistical tests are conducted simultaneously.

To test the micro-hypotheses of H4, we use the paired t -test because it is suitable for comparing the mean performance metrics between an optimized regression model with a baseline model. In our study, each performance metric (MSE and R^2) is obtained through k -fold cross-validation, generating multiple values for each metric across different folds. The paired t -test is designed to compare these paired observations, making it ideal for determining if the mean performance of our optimized models significantly deviates from the baseline models across the folds.

By using a directional (one-tailed) version of this test, we can specifically test if the MSE and the R^2 of an optimized model is significantly lower and higher (respectively) compared to a baseline model. The baseline model is different for each micro-hypothesis, as we are looking to compare the performance of a given optimized model compared to a more simple baseline model. To clarify what the baseline and optimized models are in each micro-hypothesis, we present a summary in Table 4.3.

4

Results

4.1. Data Collection and Processing

After the data collection process was completed, a total of 325 ratings from participants were collected. Each video was rated an average of 6.2 times. During the data cleaning process, we rejected 9 ratings due to poor quality. These rejections were based on clear indications that participants did not put effort into the survey. For instance, one participant repeatedly copied and pasted the same text in the appraisals and rated the same value for every single question. After removing these poor-quality ratings, the dataset was reduced to 3160 ratings (recalling that each participant rated 10 videos).

Next, we performed outlier detection on the dependent variables, which further reduced the dataset from 3160 to 2775 rows. This step was essential to ensure the reliability and validity of the subsequent analyses. Post-outlier detection, each video was rated an average of 5.59 times.

4.1.1. Exploratory Data Analysis

To understand the spread and distribution of the measurements of the robot's emotional qualities, we present the descriptive statistics of the dependent variables. Furthermore, additional plots generated during the exploratory data analysis phase, which offer deeper insights into the distribution and variability of these variables, can be found in Appendix Section Section G.1.

Table 4.1: Descriptive statistics of the dependent variables post-aggregation

	Mean	Std. Dev.	Min.	Q1	Median	Q3	Max.
Joy Intensity	1.795	0.844	0.000	1.167	1.833	2.333	4.500
Sadness Intensity	1.435	0.777	0.000	0.833	1.333	2.000	4.000
Fear Intensity	1.335	0.705	0.000	0.800	1.250	1.808	3.500
Anger Intensity	0.888	0.581	0.000	0.429	0.833	1.200	3.333
Disgust Intensity	0.525	0.333	0.000	0.333	0.500	0.667	1.667
Surprise Intensity	1.412	0.685	0.000	0.857	1.333	1.833	3.667
Pleasure	4.525	1.229	1.600	3.667	4.600	5.333	8.200
Arousal	5.041	1.195	2.000	4.200	5.000	6.000	7.667
Dominance	4.385	1.035	1.667	3.667	4.333	5.000	7.800

4.2. Correlation Analysis

The statistically significant correlations according to Spearman's rank-order correlation coefficient with the Bonferroni correction applied is presented in Table 4.2. The correlation coefficients ρ are sorted by magnitude from highest to lowest to indicate which dependent variables had the strongest correlation with each of the independent variables.

Table 4.2: Summary of significant correlations between the independent and dependent variables

Dependent Variable	Independent Variable	Spearman ρ	Corrected p -value
Joy Intensity	Blink Temperature	-0.152	0.000
	Beep Cycle Rate	0.149	0.000
	Beep Slope	0.14	0.000
	Wander Speed	0.121	0.000
	Wander Cycle Rate	-0.079	0.003
	Blink Cycle Rate	0.069	0.024
Sadness Intensity	Wander Speed	-0.264	0.000
	Beep Slope	-0.174	0.000
	Beep Cycle Rate	-0.096	0.000
Fear Intensity	Beep Pitch	0.206	0.000
	Wander Roundness	-0.19	0.000
	Wander Cycle Rate	0.188	0.000
	Wander Speed	0.119	0.000
Anger Intensity	Wander Speed	0.176	0.000
	Blink Temperature	0.156	0.000
	Wander Roundness	-0.104	0.000
Disgust Intensity	Beep Pitch	-0.079	0.002
Surprise Intensity	Wander Speed	0.297	0.000
	Beep Pitch	0.232	0.000
	Wander Roundness	-0.224	0.000
	Beep Slope	0.216	0.000
	Blink Cycle Rate	0.068	0.030
Pleasure	Wander Speed	0.147	0.000
	Wander Cycle Rate	-0.14	0.000
	Beep Slope	0.106	0.000
	Blink Temperature	-0.099	0.000
	Beep Cycle Rate	0.092	0.000
Arousal	Wander Speed	0.556	0.000
	Beep Pitch	0.202	0.000
	Wander Roundness	-0.171	0.000
	Beep Slope	0.127	0.000
	Beep Cycle Rate	0.111	0.000
Dominance	Wander Speed	0.257	0.000
	Wander Cycle Rate	-0.165	0.000
	Beep Slope	0.101	0.000
	Beep Cycle Rate	0.092	0.000
	Blink Temperature	0.086	0.000

4.3. Regression Models

As outlined in Section 3.5.8, we used the paired t -test to compare the mean performance metrics between and optimized regression model and a baseline. For each of the dependent variables, we will present a table to report the results that yielded significant differences in performance scores between the optimized and baseline models. The reported scores in these tables are the average performance metrics of the k :

models on unseen data (i.e., the testing data set). In Table 4.3, we present an overview of the definition of the optimized model and its corresponding baseline model for reference.

Table 4.3: Baseline and optimized models of each micro-hypothesis of H4

ID	Optimized Model	Baseline Model
H4.1	Linear model trained with X	Mean predictor of the dependent variable
H4.2	Linear model trained with X_{int}	Linear model trained with X
H4.3	Linear model trained with X_{gmm}	Linear model trained with X
H4.4	Linear model trained with X_{gmm_int}	Linear model trained with X_{gmm}
H4.5	Random forest model trained with X	Mean predictor of the dependent variable
H4.6	Random forest model trained with X_{int}	Random forest model trained with X
H4.7	Random forest model trained with X_{gmm}	Random forest model trained with X
H4.8	Random forest model trained with X_{gmm_int}	Random forest model trained with X_{gmm}

To assess if the regression models are overfitting, we calculate the overfitting ratio, which approximates the extent of overfitting by calculating the ratio of the training error to the testing error ($OR = MSE_{train} / MSE_{test}$). An $OR \approx 1$ indicates similar performance on both training and testing data, suggesting minimal overfitting. We will use t -tests to determine if the OR is significantly different from 1. This analysis is crucial because an overfitting ratio equal to 1 implies that the model performs equally well on both datasets, indicating no overfitting. Additionally, we will include scatterplots of the actual versus predicted values in the training and testing datasets for all models in Appendix Section G.3, offering a visual assessment method to further evaluate potential overfitting.

Finally, we will report the (at most ten) most important features of the the model that achieved the lowest MSE score in the training dataset. For simplicity, we will refer to this model as the “best” model. This could be any of the linear or random forest models trained in any of the four training datasets (X , X_{int} , X_{gmm} or X_{gmm_int}). Depending on whether the best model is a linear or random forest model, the importance of the features will be ranked by the absolute value of the estimated linear coefficients or the Gini importance. This will be done to highlight the features that are most significant in terms of how they contribute to predicting the dependent variables.

4.3.1. Joy Intensity Models

The significant results of the comparison of the performance scores between the baseline and optimized regression models for joy intensity are reported in Table 4.4.

Table 4.4: Performance scores between the optimized and baseline models for joy intensity

Model Type	Dataset	Score Type	Optimized	Baseline	t -statistic	p -value
Linear	X	MSE	0.667	0.711	-2.955	0.021
		R^2	0.056	-0.005	2.996	0.020
	X_{gmm}	MSE	0.645	0.667	-3.432	0.013
		R^2	0.087	0.056	3.339	0.014
Random Forest	X	MSE	0.624	0.711	-6.879	0.001
		R^2	0.118	-0.005	6.367	0.002

These results indicate that both the linear and random forest model trained with the nine modal parameters only (X) significantly outperformed a mean predictor of the perceived average joy intensity on unseen data. Moreover, the linear model trained with X_{gmm} exhibited better performance than the one trained with X . The highest predictive accuracy on unseen data was achieved by the random forest model trained with X .

However, the tests to determine if the OR of the joy regression models is significantly different from 1 reveal that only the linear models trained with the datasets X , X_{int} or X_{gmm} did not significantly overfit the data:

Table 4.5: Results of the overfitting ratio tests for the regression models of joy intensity

Model Type	Dataset	Average Training MSE	Average OR	t -statistic	p -value
Linear	X	0.654	0.924	-2.144	0.099
	X_{int}	0.614	0.925	-1.797	0.147
	X_{gmm}	0.628	0.946	-1.287	0.268

The ten most important features of the best joy intensity model (i.e., the model that achieved the highest predictive accuracy on unseen data) are illustrated in Figure 4.1. As is visible, the most important feature of this model was the `blinkTemperature`, which coincides with the results of the Spearman ρ correlation analysis reported in Table 4.2.

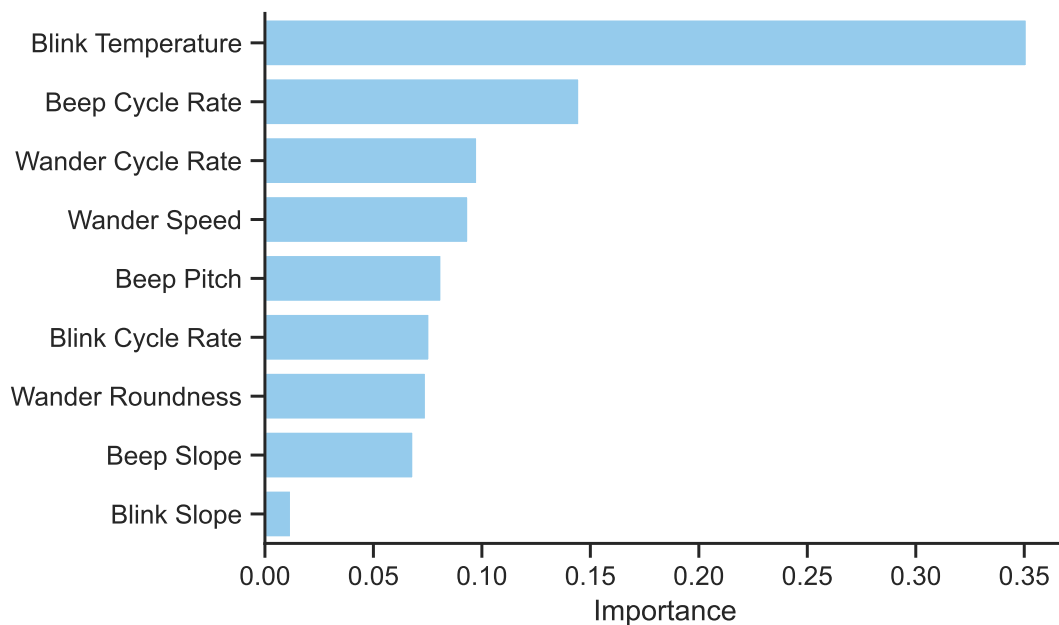


Figure 4.1: Most important features of the best regression model for joy intensity

4.3.2. Sadness Intensity Models

The significant results of the comparison of the performance scores between the baseline and optimized regression models for sadness intensity are reported in Table 4.6.

Table 4.6: Performance scores between the optimized and baseline models for sadness intensity

Model Type	Dataset	Score Type	Optimized	Baseline	t -statistic	p -value
Linear	X	MSE	0.538	0.602	-5.662	0.002
		R^2	0.105	-0.006	4.841	0.004
	X_{gmm}	MSE	0.464	0.538	-6.299	0.002
		R^2	0.229	0.105	6.771	0.001
Random Forest	X	MSE	0.476	0.602	-6.13	0.002
		R^2	0.210	-0.006	5.813	0.002

Continued on the next page

Table 4.6: Continued from previous page

Model Type	Dataset	Score Type	Optimized	Baseline	t -statistic	p -value
Random Forest	X_{gmm}	MSE	0.463	0.476	-3.269	0.015
		R^2	0.231	0.210	3.493	0.013

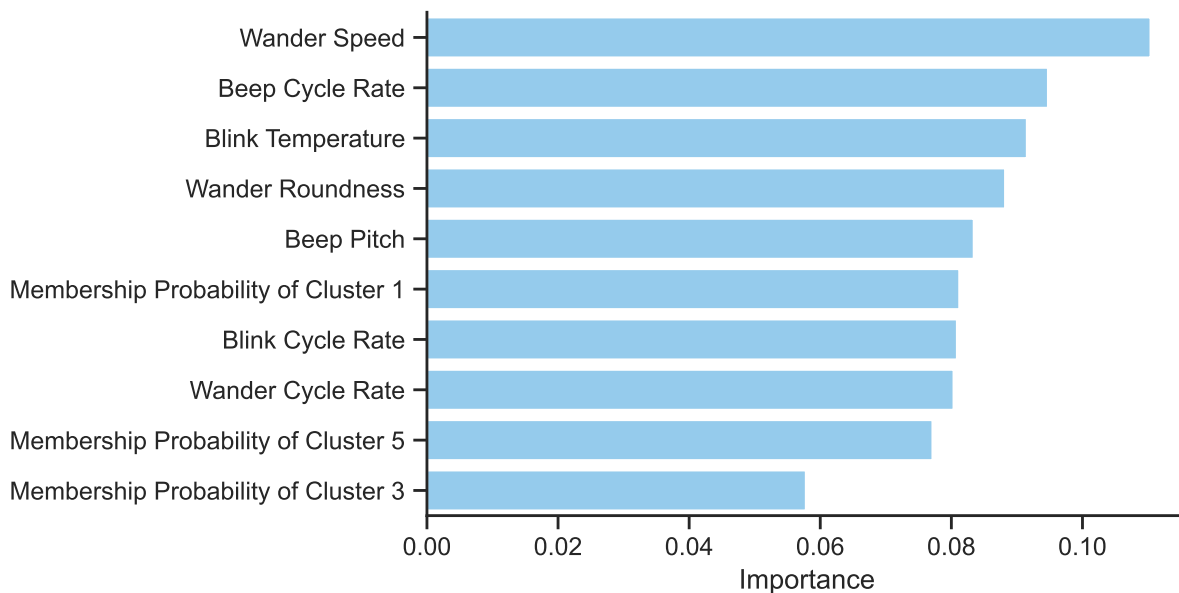
Similar to the results reported in Table 4.4, these results indicate that both the linear and random forest model trained with the nine modal parameters only (X) significantly outperformed a mean predictor of the perceived average sadness intensity on unseen data. Moreover, both the linear and the random forest model trained with X_{gmm} exhibited better performance than the one trained with X . The highest predictive accuracy on unseen data was achieved by the random forest model trained with X_{gmm} .

However, the tests to determine if the OR of the sadness regression models also show that only the linear models trained with any of the datasets did not significantly overfit the data:

Table 4.7: Results of the overfitting ratio tests for the regression models of sadness intensity

Model Type	Dataset	Average Training MSE	Average OR	t -statistic	p -value
Linear	X	0.533	0.901	-1.383	0.239
	X_{int}	0.508	0.976	-0.224	0.834
	X_{gmm}	0.454	0.873	-1.395	0.236
	X_{gmm}_{int}	0.387	0.872	-1.263	0.275

The ten most important features of the best sadness intensity model are illustrated in Figure 4.2. As is visible, the most important feature of this model was the `wanderSpeed`, which coincides with the results of the Spearman ρ correlation analysis reported in Table 4.2.

**Figure 4.2:** Most important features of the best regression model for sadness intensity

4.3.3. Fear Intensity Models

The significant results of the comparison of the performance scores between the baseline and optimized regression models for fear intensity are reported in Table 4.8.

Table 4.8: Performance scores between the optimized and baseline models for fear intensity

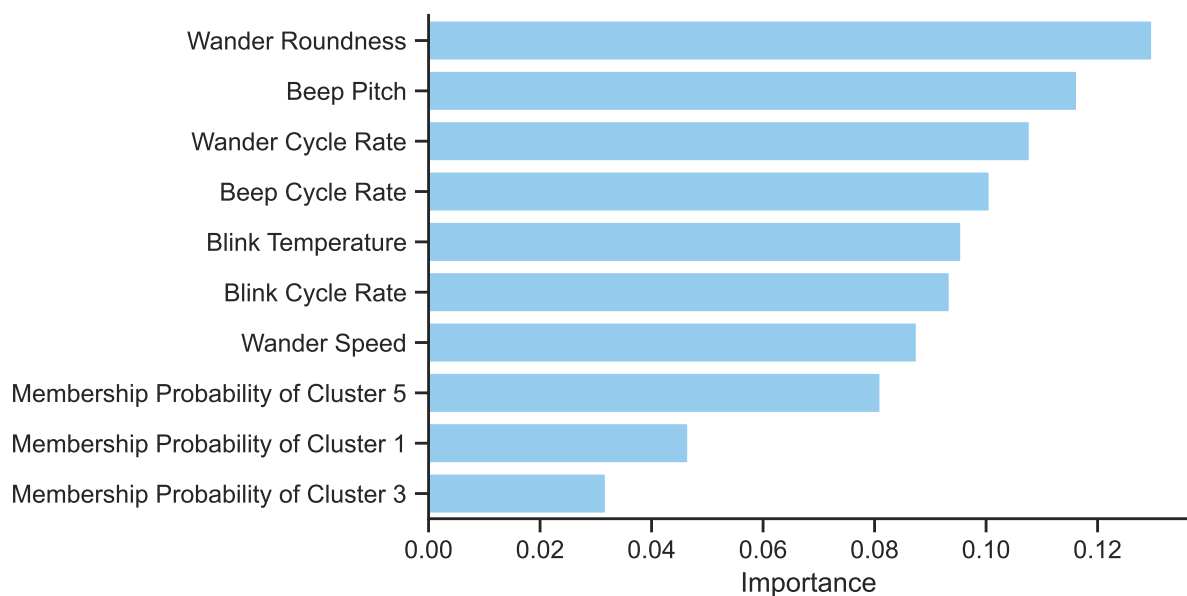
Model Type	Dataset	Score Type	Optimized	Baseline	<i>t</i> -statistic	<i>p</i> -value
Linear	X	MSE	0.448	0.497	-3.151	0.017
		R^2	0.083	-0.012	3.459	0.013
	X_gmm	MSE	0.438	0.448	-2.186	0.047
		R^2	0.104	0.083	2.209	0.046
Random Forest	X	MSE	0.447	0.497	-3.82	0.009
		R^2	0.087	-0.012	4.012	0.008
	X_gmm	MSE	0.421	0.447	-7.566	0.001
		R^2	0.139	0.087	10.873	0.000

Exactly as in the sadness intensity models, these results indicate that both the linear and random forest model trained with the nine modal parameters only (X) significantly outperformed a mean predictor of the perceived average fear intensity on unseen data. Moreover, both the linear and the random forest model trained with X_gmm exhibited better performance than the one trained with X. The highest predictive accuracy on unseen data was achieved by the random forest model trained with X_gmm.

The tests to determine if the *OR* of the fear regression models also show that only the linear models trained with any of the datasets did not significantly overfit the data:

Table 4.9: Results of the overfitting ratio tests for the regression models of fear intensity

Model Type	Dataset	Average Training MSE	Average <i>OR</i>	<i>t</i> -statistic	<i>p</i> -value
Linear	X	0.432	0.907	-0.943	0.399
	X_int	0.41	0.948	-0.542	0.617
	X_gmm	0.415	0.96	-0.396	0.712
	X_gmm_int	0.349	0.826	-2.035	0.112

**Figure 4.3:** Most important features of the best regression model for fear intensity

The ten most important features of the best fear intensity model are illustrated in Figure 4.3. As is visible, the most important feature of this model was the `wanderRoundness`, which appears as the second strongest Spearman ρ for fear intensity.

4.3.4. Anger Intensity Models

The significant results of the comparison of the performance scores between the baseline and optimized regression models for anger intensity are reported in Table 4.10.

Table 4.10: Performance scores between the optimized and baseline models for anger intensity

Model Type	Dataset	Score Type	Optimized	Baseline	<i>t</i> -statistic	<i>p</i> -value
Linear	X	MSE	0.318	0.337	-3.185	0.017
		R^2	0.055	-0.005	2.908	0.022
Random Forest	X	MSE	0.296	0.337	-7.768	0.001
		R^2	0.122	-0.005	5.533	0.003
	X_gmm	MSE	0.288	0.296	-2.325	0.04
		R^2	0.144	0.122	2.456	0.035

These results indicate that both the linear and random forest model trained with the nine modal parameters only (X) significantly outperformed a mean predictor of the perceived average anger intensity on unseen data. Additionally, the random forest model trained with X_gmm exhibited better performance than the one trained with X. The highest predictive accuracy on unseen data was also achieved by the random forest model trained with X_gmm.

However, the tests to determine if the *OR* of the anger regression models show that only the linear models trained with any of the datasets did not significantly overfit the data:

Table 4.11: Results of the overfitting ratio tests for the regression models of anger intensity

Model Type	Dataset	Average Training MSE	Average <i>OR</i>	<i>t</i> -statistic	<i>p</i> -value
Linear	X	0.307	0.928	-1.026	0.363
	X_int	0.299	0.969	-0.342	0.749
	X_gmm	0.3	0.97	-0.341	0.751
	X_gmm_int	0.255	0.83	-2.032	0.112

The ten most important features of the best anger intensity model are illustrated in Figure 4.4. As is visible, the most important feature of this model was the `blinkTemperature`, which also coincides with the second strongest Spearman ρ for anger intensity.

4.3.5. Disgust Intensity Models

For disgust intensity, there were no models that significantly outperformed the mean predictor on unseen data. The only model that achieved marginally significant performance metrics is reported in Table 4.12.

Table 4.12: Performance scores between the optimized and baseline models for disgust intensity

Model Type	Dataset	Score Type	Optimized	Baseline	<i>t</i> -statistic	<i>p</i> -value
Random Forest	X_gmm	MSE	0.109	0.110	-2.005	0.058
		R^2	0.003	-0.005	1.962	0.061

Since none of the models achieved significant results, it may be concluded that it is not possible to accurately predict the perceived disgust intensity with any of the engineered features. For that reason,

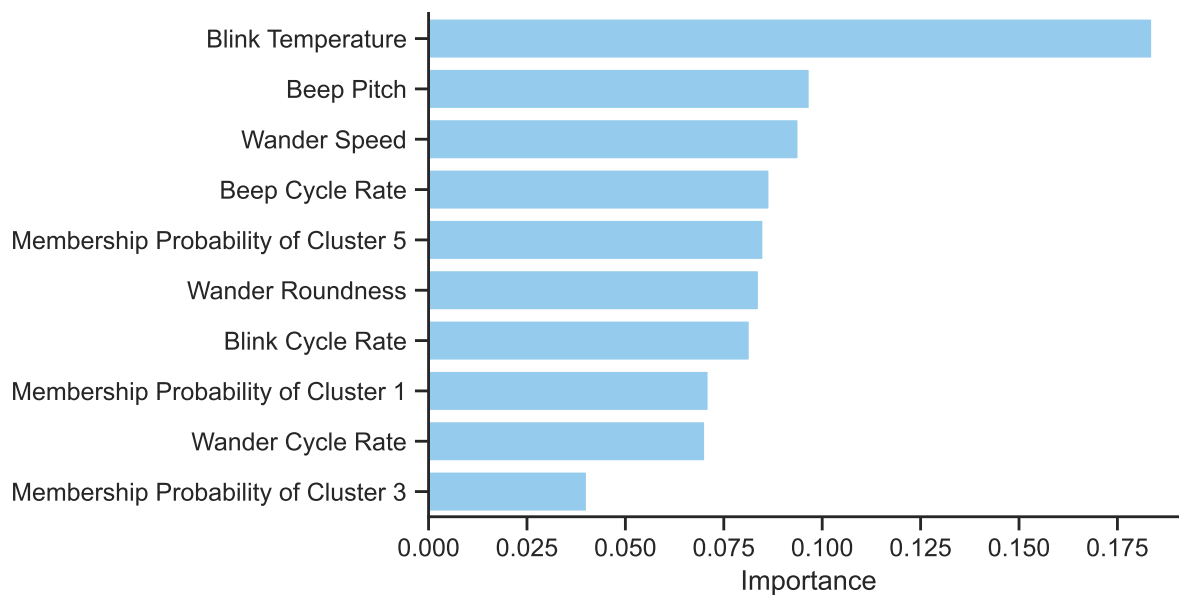


Figure 4.4: Most important features of the best regression model for anger intensity

we do not report which models did not significantly overfit the data, nor the most important features of the disgust intensity model that achieved the highest predictive accuracy on unseen data.

4.3.6. Surprise Intensity Models

The significant results of the comparison of the performance scores between the baseline and optimized regression models for surprise intensity are reported in Table 4.13.

Table 4.13: Performance scores between the optimized and baseline models for surprise intensity

Model Type	Dataset	Score Type	Optimized	Baseline	<i>t</i> -statistic	<i>p</i> -value
Linear	X	MSE	0.372	0.468	-3.443	0.013
		R^2	0.195	-0.01	3.356	0.014
Random Forest	X	MSE	0.367	0.468	-5.012	0.004
		R^2	0.206	-0.010	5.036	0.004

These results indicate that both the linear and random forest model trained with the nine modal parameters only (X) significantly outperformed a mean predictor of the perceived average surprise intensity on unseen data. Additionally, training the linear or random forest models with any of the other engineered features did not yield significantly better results. The highest predictive accuracy on unseen data was therefore achieved by the random forest model trained with X.

However, the tests to determine if the *OR* of the surprise regression models is significantly different from 1 reveal that only the linear models trained with the datasets X_{int} , X_{gmm} or X_{gmm_int} did not significantly overfit the data:

Table 4.14: Results of the overfitting ratio tests for the regression models of surprise intensity

Model Type	Dataset	Average Training MSE	Average <i>OR</i>	<i>t</i> -statistic	<i>p</i> -value
Linear	X_{int}	0.349	0.968	-0.339	0.752
	X_{gmm}	0.342	0.948	-0.574	0.597
	X_{gmm_int}	0.303	0.875	-1.705	0.163

The ten most important features of the best surprise intensity model are illustrated in Figure 4.5. As is visible, the most important feature of this model was the `wanderRoundness`, which surprisingly only coincides with the third strongest Spearman ρ for surprise intensity.

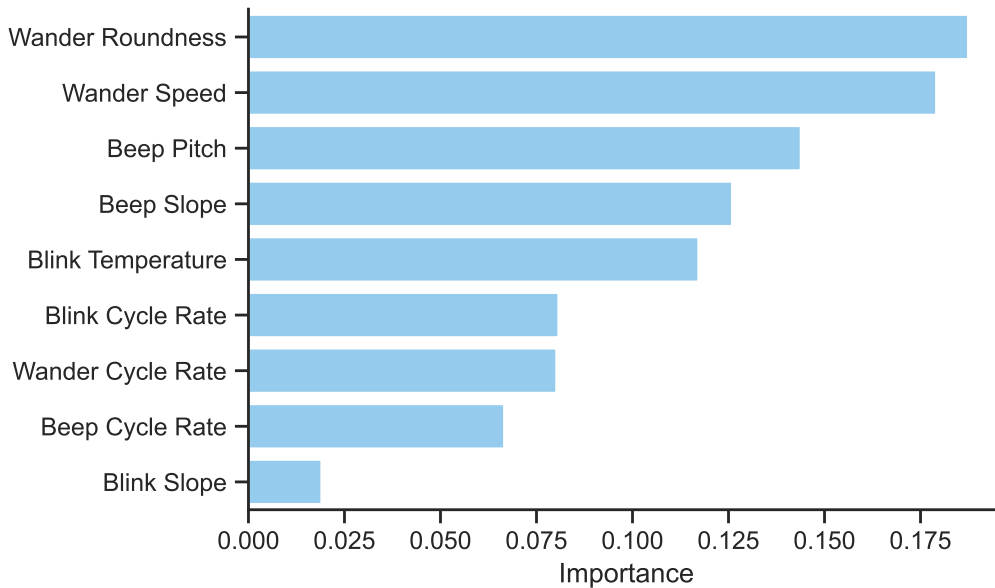


Figure 4.5: Most important features of the best regression model for surprise intensity

4.3.7. Pleasure Models

The significant results of the comparison of the performance scores between the baseline and optimized regression models for pleasure are reported in Table 4.15.

Table 4.15: Performance scores between the optimized and baseline models for pleasure

Model Type	Dataset	Score Type	Optimized	Baseline	<i>t</i> -statistic	<i>p</i> -value
Linear	X_{gmm}	MSE	1.38	1.456	-2.219	0.045
		R^2	0.076	0.029	2.281	0.042
	X_{int}	MSE	1.39	1.456	-3.113	0.018
		R^2	0.072	0.029	3.459	0.013
Random Forest	X	MSE	1.327	1.507	-4.614	0.005
		R^2	0.114	-0.006	4.451	0.006
	X_{int}	MSE	1.308	1.327	-4.842	0.004
		R^2	0.127	0.114	4.184	0.007

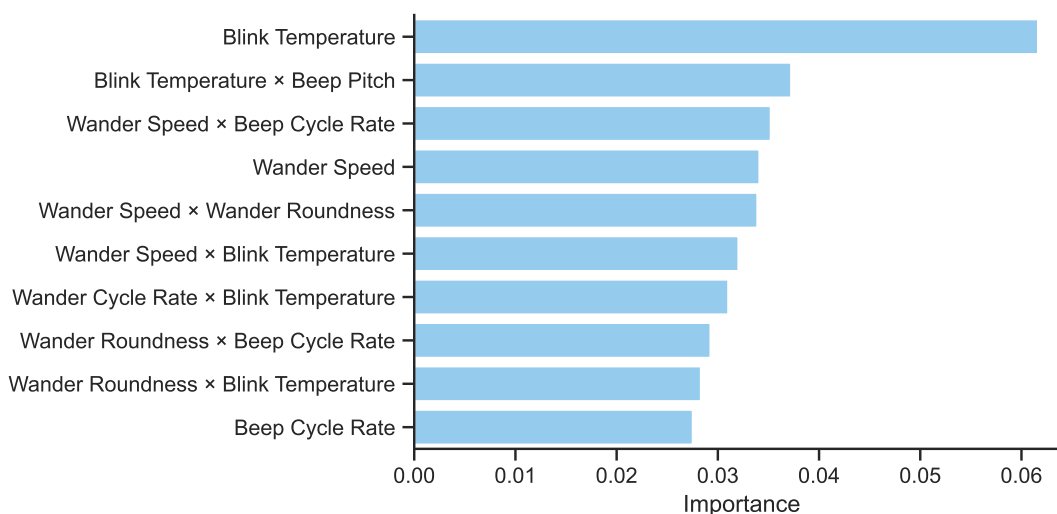
These results indicate that only the random forest model trained with the nine modal parameters (X) significantly outperformed a mean predictor of the perceived average pleasure on unseen data. However, the linear model trained with X_{gmm} outperformed the mean predictor. Additionally, training both the linear and random forest models with the nine modal parameters and their interaction terms X_{int} yielded significantly better results. The highest predictive accuracy on unseen data was achieved by the random forest model trained with X_{int} .

However, the tests to determine if the *OR* of the pleasure regression models show that only the linear models trained with any of the datasets did not significantly overfit the data:

Table 4.16: Results of the overfitting ratio tests for the regression models of pleasure

Model Type	Dataset	Average Training MSE	Average <i>OR</i>	<i>t</i> -statistic	<i>p</i> -value
Linear	X	1.397	0.938	-1.031	0.361
	X_int	1.322	0.924	-1.048	0.354
	X_gmm	1.308	0.915	-1.157	0.312
	X_gmm_int	1.137	0.837	-2.463	0.07

The ten most important features of the best pleasure model are illustrated in Figure 4.6. As is visible, the most important feature of this model was the `blinkTemperature`, which coincides with the fourth strongest Spearman ρ for pleasure.

**Figure 4.6:** Most important features of the best regression model for pleasure

4.3.8. Arousal Models

The significant results of the comparison of the performance scores between the baseline and optimized regression models for arousal are reported in Table 4.17.

Table 4.17: Performance scores between the optimized and baseline models for arousal

Model Type	Dataset	Score Type	Optimized	Baseline	<i>t</i> -statistic	<i>p</i> -value
Linear	X	MSE	0.902	1.425	-4.321	0.006
		R^2	0.344	-0.018	4.263	0.007
	X_gmm	MSE	0.785	0.902	-8.605	0.001
		R^2	0.429	0.344	6.873	0.001
Random Forest	X	MSE	0.901	1.425	-4.481	0.005
		R^2	0.345	-0.018	4.462	0.006

These results indicate that both the linear and the random forest models trained with the nine modal parameters (X) significantly outperformed a mean predictor of the perceived average arousal on unseen data. Additionally, the linear model trained with the nine modal parameters and the six GMM cluster membership probabilities (X_gmm) yielded the highest predictive accuracy on unseen data.

Additionally, the tests to determine if the *OR* of the arousal regression models show that this model did not significantly overfit the data:

Table 4.18: Results of the overfitting ratio tests for the regression models of arousal

Model Type	Dataset	Average Training MSE	Average <i>OR</i>	<i>t</i> -statistic	<i>p</i> -value
Linear	X_int	0.817	0.955	-0.386	0.719
	X_gmm	0.736	0.863	-1.253	0.278
	X_gmm_int	0.657	0.895	-0.869	0.434

Since the best model for arousal, is a linear model, instead of reporting the feature importance, we will present the most significant coefficients in Table 4.19. As is visible, the most significant feature of this model was the `wanderSpeed`, which coincides with the results of the strongest Spearman ρ correlation analysis in Table 4.2.

Table 4.19: Most significant features of the best regression model for arousal

Feature	Coefficient	<i>p</i> -value
Wander Speed	0.652	0.000
Membership Probability of Cluster 4	0.503	0.000
Beep Slope	-0.375	0.005
Beep Pitch	0.252	0.000
Wander Roundness	-0.195	0.001
Membership Probability of Cluster 1	-0.165	0.001
Membership Probability of Cluster 3	-0.151	0.006
Membership Probability of Cluster 2	-0.044	0.000

4.3.9. Dominance Models

The significant results of the comparison of the performance scores between the baseline and optimized regression models for dominance are reported in Table 4.20.

Table 4.20: Performance scores between the optimized and baseline models for dominance

Model Type	Dataset	Score Type	Optimized	Baseline	<i>t</i> -statistic	<i>p</i> -value
Linear	x	MSE	0.998	1.069	-3.651	0.011
		R^2	0.058	-0.008	3.710	0.010
Random Forest	x	MSE	0.960	1.069	-3.217	0.016
		R^2	0.088	-0.008	3.150	0.017

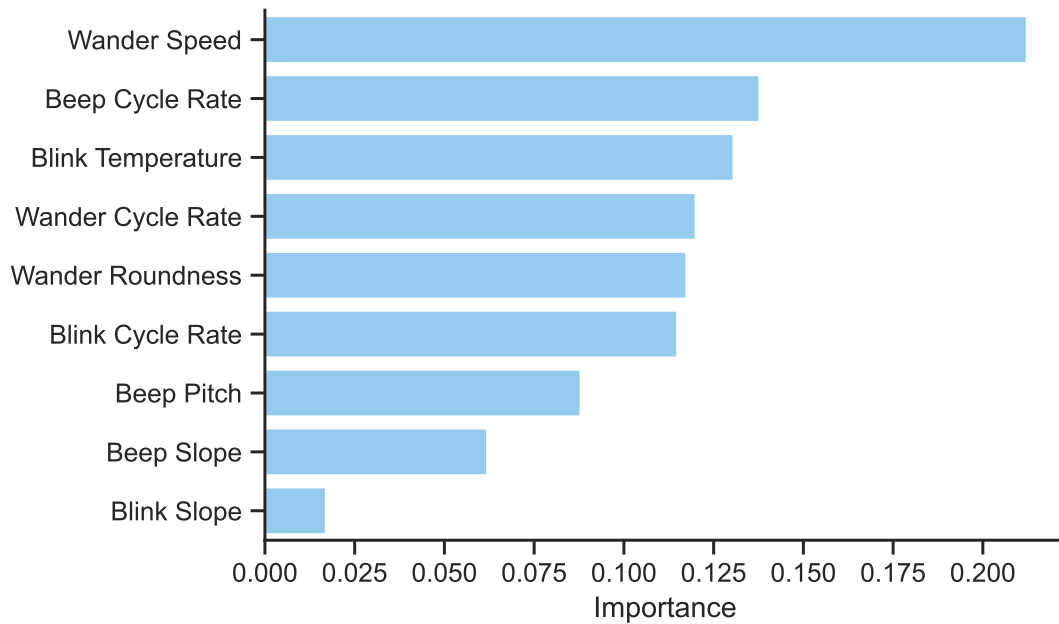
These results indicate that both the linear and the random forest models trained with the nine modal parameters (x) significantly outperformed a mean predictor of the perceived average dominance on unseen data. Additionally, training the linear or random forest models with any of the other engineered features did not yield significantly better results. The highest predictive accuracy on unseen data was therefore achieved by the random forest model trained with x .

However, the tests to determine if the *OR* of the dominance regression models show that only the linear models trained with any of the datasets did not significantly overfit the data:

Table 4.21: Results of the overfitting ratio tests for the regression models of dominance

Model Type	Dataset	Average Training MSE	Average <i>OR</i>	<i>t</i> -statistic	<i>p</i> -value
Linear	X	0.948	0.906	-1.288	0.267
	X_int	0.91	0.93	-1.043	0.356
	X_gmm	0.899	0.919	-1.21	0.293
	X_gmm_int	0.805	0.865	-2.761	0.051

The ten most important features of the best dominance model are illustrated in Figure 4.7. As is visible, the most important feature of this model was the `wanderSpeed`, which coincides with the results of the strongest Spearman ρ correlation analysis in Table 4.2.

**Figure 4.7:** Most important features of the best regression model for dominance

5

Discussion

The primary objective of this study was to explore the relationship between specific parameters within the modalities of motion, light, and sound, and the emotional perceptions attributed to a non-humanoid, faceless robot. A key aspect of this research was to determine whether it is possible to predict the emotional qualities perceived in the robot's behaviors based on variations in these parameters.

By manipulating factors such as speed, light intensity, and sound pitch across the robot's three primary behavior modalities (motion, light, and sound), we aimed to establish a quantitative link between these input parameters and the emotional qualities they convey.

5.1. Research Questions

This study seeks to address the main research question (MRQ): Can variations in motion, light, and sound parameters of the behaviors of a non-humanoid, faceless robot influence the perceived emotional qualities of those behaviors? To comprehensively answer this MRQ, we formulated four specific sub-questions to guide our investigation:

1. **RQ1:** How do variations in motion parameters (speed, roundness, cycle rate) influence the perceived emotional qualities of the robot's behavior?
2. **RQ2:** How do variations in light parameters (light temperature, change in brightness, tempo) influence the perceived emotional qualities of the robot's behavior?
3. **RQ3:** How do variations in sound parameters (pitch, intonation, tempo) influence the perceived emotional qualities of the robot's behavior?
4. **RQ4:** Can regression models, trained with features derived from the input parameters of the robot's modal behaviors, effectively predict the perceived emotional qualities of those behaviors?

To answer these sub-questions, we formulated hypotheses and conducted tests to gather evidence to provide answers. In the following sections, we will critically discuss the findings in relation to each research question and compare them with the work of others introduced in Chapter 2.

5.1.1. RQ1: Motion Parameters and Emotional Perception

The analysis of the correlations between motion parameters and perceived emotional qualities revealed several significant relationships that align with theoretical predictions. Higher wander speeds are strongly associated with increased arousal ($\rho = 0.556$, $p \approx 0$), confirming that faster movements are perceived as more arousing. Additionally, there are positive correlations between higher speeds and perceived surprise ($\rho = 0.297$, $p \approx 0$), dominance ($\rho = 0.257$, $p \approx 0$), and pleasure ($\rho = 0.147$, $p \approx 0$), suggesting that faster speeds contribute to higher arousal, surprise, dominance, and positive pleasure. These findings support the notion that high speed is perceived as more positive and arousing, consistent with theoretical expectations.

Furthermore, rounded movement patterns are associated with positive pleasure, as indicated by a weak but significant positive correlation with mean pleasure ($\rho = 0.062$, $p = 0.093$). The negative correlation between wander roundness and arousal ($\rho = -0.171$, $p \approx 0$) also aligns with the theory that sharp and angular movements convey high arousal. Although the effect on pleasure is less pronounced, the relationship between rounded movements and positive pleasure is evident.

The analysis also confirms that faster movements can be associated with joy, as evidenced by the weak positive correlation between wander speed and joy intensity ($\rho = 0.121$, $p \approx 0$). Additionally, higher

speeds are linked with increased perceptions of anger ($\rho = 0.176, p \approx 0$), supporting the idea that fast, jerky movements can convey anger.

Lastly, the prediction that small and slow movements are attributed to sadness and fear is partially supported. While there is a negative correlation between speed and sadness ($\rho = -0.264, p \approx 0$), indicating slower movements are associated with sadness, the correlation for fear is weakly positive ($\rho = 0.119, p \approx 0$), which does not strongly confirm this prediction.

5.1.2. RQ2: Light Parameters and Emotional Perception

The analysis of the correlations between light parameters and perceived emotional qualities also revealed several significant relationships. The blink temperature shows a weak positive correlation with anger intensity ($\rho = 0.156, p \approx 0$), indicating that warmer colors might convey negative emotions such as anger. Conversely, blink temperature is negatively correlated with joy intensity ($\rho = -0.152, p \approx 0$) and pleasure ($\rho = -0.099, p \approx 0$), suggesting that warmer colors are associated with lower perceived joy and pleasure, which aligns with the theory that warm colors convey negative pleasure. There is also a weak positive correlation between blink temperature and dominance ($\rho = 0.086, p \approx 0$), indicating that warmer colors might be perceived as more dominant. However, the correlation between blink temperature and arousal is not significant ($\rho = 0.063, p = 0.08$), nor are the correlations with surprise, fear, sadness, and disgust.

For blink slope, no significant correlations were found with any of the emotional qualities, indicating that changes in brightness do not significantly influence the perceived emotions in the context of this study.

Blink cycle rate, however, shows a weak positive correlation with joy intensity ($\rho = 0.069, p = 0.024$) and surprise intensity ($\rho = 0.068, p = 0.03$), suggesting that higher blinking frequency might slightly increase the perceptions of joy and surprise. This partially confirms the theoretical prediction that there is a positive correlation between arousal and blinking frequency, although the direct correlation with arousal itself is not significant ($\rho = 0.027, p = 1.0$).

5.1.3. RQ3: Sound Parameters and Emotional Perception

The analysis of the correlations between sound parameters and perceived emotional also revealed some interesting findings. The beep pitch shows a moderate positive correlation with surprise intensity ($\rho = 0.232, p \approx 0$) and fear intensity ($\rho = 0.206, p \approx 0$), indicating that higher pitches are associated with increased perceptions of surprise and fear. Additionally, there is a positive correlation between beep pitch and arousal ($\rho = 0.202, p \approx 0$), suggesting that higher pitches increase perceived arousal. Interestingly, beep pitch is negatively correlated with disgust intensity ($\rho = -0.079, p = 0.002$), meaning higher pitches might slightly reduce perceived disgust. Other correlations with beep pitch, including joy, dominance, pleasure, sadness, and anger, were not significant.

The beep slope, which represents the intonation change over time, shows significant correlations with several emotional qualities. There is a positive correlation between beep slope and surprise intensity ($\rho = 0.216, p \approx 0$), joy intensity ($\rho = 0.140, p \approx 0$), arousal ($\rho = 0.127, p \approx 0$), pleasure ($\rho = 0.106, p \approx 0$), and dominance ($\rho = 0.101, p \approx 0$). These findings suggest that rising intonation is perceived as more surprising, joyful, arousing, pleasurable, and dominant. Conversely, beep slope is negatively correlated with sadness intensity ($\rho = -0.174, p \approx 0$), supporting the theory that falling intonation is associated with sadness. No significant correlations were found between beep slope and anger, fear, or disgust.

The beep cycle rate shows a significant positive correlation with joy intensity ($\rho = 0.149, p \approx 0$) and arousal ($\rho = 0.111, p \approx 0$), indicating that faster beep cycles are perceived as more joyful and arousing. Additionally, beep cycle rate is positively correlated with pleasure ($\rho = 0.092, p \approx 0$) and dominance ($\rho = 0.092, p \approx 0$), suggesting that higher frequencies can increase perceptions of pleasure and dominance. However, no significant correlation was found between beep cycle rate and surprise, disgust, anger, or fear.

These results suggest that sounds are good at conveying certain negative emotions, such as fear and sadness, as indicated by the correlations with the beep pitch and slope. However, the associations with positive emotions, particularly through beep slope and cycle rate, suggest that sounds can also convey positive emotions effectively.

5.1.4. RQ4: Predictive Modeling of Emotional Perception

To evaluate the potential of machine learning models in predicting emotional qualities based on the robot's behavioral parameters, we trained and tested linear regression models and random forest regressors.

Additionally, we engineered additional features derived from these behavioral parameters such as interaction terms and GMM clustering probabilities to investigate if this could potentially improve the models' predictive accuracy. Indeed, the hypotheses for RQ4 aims to compare the MSE and R^2 between the baseline and optimized models on unseen data to assess model generalizability and performance.

Categorical Emotions

The results for joy intensity indicate that both linear and random forest models trained with the nine modal parameters (X) significantly outperformed a mean predictor on unseen data, suggesting that these parameters are somewhat effective predictors. Incorporating GMM cluster probabilities further improved the linear model's performance. The random forest model with X achieved the highest accuracy but had a limited R^2 of 0.118, indicating it explained little variance. Overfitting ratio analysis showed that only the linear models trained with X , X_{gmm_int} , and X_{gmm} did not significantly overfit, suggesting better generalization compared to the random forest model, which likely overfitted. Feature importance analysis highlights the blink temperature, beep cycle rate, wander cycle rate, and wander speed as key predictors of joy intensity, which align with the findings of the correlation analysis.

Additionally, the regression model results for sadness intensity show that both linear and random forest models trained with the nine modal parameters (X) also significantly outperformed a mean predictor on unseen data. The linear model trained with X_{gmm} further improved performance. The random forest model with X_{gmm} achieved the best predictive accuracy with an MSE of 0.463, but the OR analysis suggests this model might be overfitting. Feature importance analysis identified wander speed, beep cycle rate, blink temperature, and wander roundness as key predictors. All of these features align with our findings from the correlation analysis as well, which support the bottom-up approach and show how these parameters influence perceived sadness intensity. Additionally, the cluster membership probabilities of clusters 1, 3 and 5 fell within the ten most important features, which demonstrates the added value of engineering the features using the GMM.

The results for fear intensity, similarly to the ones of sadness intensity, indicate that both linear and random forest models trained with the nine modal parameters (X) significantly outperformed a mean predictor on unseen data. Incorporating GMM cluster probabilities further enhanced the linear model's performance. The random forest model with X_{gmm} achieved the best predictive accuracy with an MSE of 0.421. However, the R^2 values indicate that these models explain only a modest portion of the variance in fear intensity, with the highest R^2 being 0.139 for the random forest model, reflecting its limited explanatory power. Furthermore, overfitting analysis revealed that while linear models did not significantly overfit, the random forest model exhibited signs of overfitting. Feature importance analysis identified wander roundness, beep pitch, wander cycle rate, and beep cycle rate as key predictors. Additionally, the cluster membership probabilities of clusters 1, 3 and 5 also fell within the ten most important features.

Similarly to the other categorical emotion intensities, the regression results for anger intensity show that both linear and random forest models trained with the nine modal parameters (X) significantly outperformed a mean predictor on unseen data. The random forest model with X achieved the highest accuracy with an MSE of 0.296 and an R^2 of 0.122, while incorporating GMM cluster probabilities further improved the performance to an MSE of 0.288 and R^2 of 0.144. However, the R^2 values reveal a limitation in explaining the variance of anger intensity, suggesting that the models capture only part of the complexity. Additionally, the overfitting ratio analysis indicate that the random forest model with X_{gmm} might overfit the data. The feature importance analysis identified blink temperature, beep pitch, wander speed, and beep cycle rate as key predictors, which closely aligns with our results of the Spearman correlation analysis.

The results for disgust intensity reveal a significant anomaly, as none of the models significantly outperformed the mean predictor on unseen data. The only model that showed marginally significant performance was the random forest trained with the X_{gmm} dataset, achieving an MSE of 0.109 and an R^2 of 0.003, but these results are still far from satisfactory. It is not surprising that disgust intensity was poorly predicted, given that disgust was scarcely identified in the robot's behavior, as it is not a typical emotion for a non-humanoid, faceless robot to convey. The descriptive statistics underscore this, with a mean disgust intensity of 0.525 and a high standard deviation of 0.333, indicating inconsistent perception among participants. The poor performance of the models suggests that the engineered features were insufficient to capture the nuances of disgust intensity. Consequently, it is concluded that predicting perceived disgust intensity with the given features is not feasible, and therefore, we do not report on overfitting for non-significant models or identify the most important features for the marginally significant model.

The results for surprise intensity, however, indicate that both linear and random forest models trained with the nine modal parameters (X) significantly outperformed a mean predictor on unseen data, demonstrating the predictive utility of these parameters. The linear model achieved an MSE of 0.372 and an R^2 of 0.195, while the random forest model slightly improved on this with an MSE of 0.367 and an R^2 of 0.206. Despite these improvements, the R^2 values suggest that a substantial portion of the variance in surprise intensity remains unexplained, highlighting a limitation in the models' explanatory power. Overfitting analysis indicates that while the random forest model performed well, the linear models did not significantly overfit. Feature importance analysis identified wander roundness, wander speed, beep pitch, and beep slope as key predictors, which also closely aligns with the Spearman correlation analysis.

Emotional Dimensions

For the dimensional emotional qualities, the results were similar. The results for pleasure intensity indicate modest improvements in predictive performance, with both linear models trained with X_{gmm} and X_{int} showing significant improvements over their baselines. The random forest model with X_{int} achieved the best accuracy, with an MSE of 1.308 and an R^2 of 0.127, suggesting some explanatory power but leaving much variance unexplained. The modest R^2 values highlight a limitation in the models' ability to capture the full complexity of pleasure intensity. Overfitting analysis reveals that the linear models did not significantly overfit, while the random forest models, despite better performance, may still exhibit overfitting. Feature importance analysis for the best model identified blink temperature, wander speed, and beep cycle rate, along with several interaction terms, as key predictors.

The results for arousal indicate that the linear model trained with X_{gmm} achieved the best performance on unseen data, with an MSE of 0.785 and an R^2 of 0.429, significantly outperforming the baseline mean predictor. The most important of all predictors was the wander speed, with a high coefficient of 0.652. This demonstrates that increasing speed will lead to stronger predicted values of arousal. Moreover, the OR analysis confirms that this model does not significantly overfit, demonstrating good generalization. Despite this, the relatively low R^2 value indicates that a significant portion of the variance in arousal remains unexplained, suggesting the need for additional features or further refinement to fully capture arousal intensity's complexity.

Finally, the results for dominance indicate that both linear and random forest models trained with the nine modal parameters (X) significantly outperformed a mean predictor on unseen data. The random forest model achieved an MSE of 0.96 and an R^2 of 0.088, while the linear model had an MSE of 0.998 and an R^2 of 0.058. Despite these improvements, the R^2 values are relatively low, indicating that a substantial portion of the variance in dominance remains unexplained. Overfitting analysis revealed that the linear models did not significantly overfit, with OR values not significantly different from 1. Feature importance analysis for the random forest model identified wander speed, beep cycle rate, and blink temperature as the most influential predictors, with wander speed having the highest importance. These findings suggest that while the models can predict dominance to some extent, their explanatory power is limited.

Overall, the X_{gmm} or X datasets led to better results in the regression models of the categorical intensities, while the X_{int} dataset performed better in predicting the dimensional qualities (PAD). This indicates that feature engineering was useful, yet the models demonstrated the benefits of parsimony, as none achieved better performance with the X_{gmm_int} dataset. A recommendation for future work could be to focus on engineering features that capture data patterns more effectively to maximize performance while minimizing overfitting.

5.2. Limitations

Our study has several limitations that need to be acknowledged. First, the final dataset included 3160 ratings, which, after outlier removal, was reduced to 2775 ratings. While this sample size can be considered as sufficient for analysis, a larger sample size could provide even more robust results and enhance the generalizability of the findings. Second, the study focused on a specific appearance-constrained robot, and the findings may not generalize to other appearance-constrained robots. Although the behaviors and emotional perceptions studied may have broader applicability, extensive research would be needed to confirm this across different robotic designs and contexts.

Moreover, let us recall that emotional perception is inherently subjective, and individual differences in emotional interpretation could significantly influence the results. Factors such as age, gender, cultural background, and personal experiences can all affect how emotions are perceived, adding variability to the

data that may not be fully captured in this study. For this reason, the aggregation of dependent variables by video to reduce noise may have oversimplified the data, potentially obscuring subtle variations in emotional perception. While this step aimed to enhance the robustness of our findings, it may have inadvertently masked finer nuances in the emotional responses of participants.

The descriptive statistics of the aggregated dependent variables provide key insights into the limitations of this study, particularly regarding the perception of emotional qualities conveyed by the robot's behaviors. The emotional intensities tended to be right-skewed, indicating that participants generally perceived these emotions as weaker. For example, joy has a mean intensity of 1.795 (std = 0.844) with a maximum of 4.5, and sadness has a mean intensity of 1.435 (std = 0.777) with a maximum of 4.0. Anger and fear show low mean intensities of 0.888 (std = 0.581) and 1.335 (std = 0.705), respectively, while disgust has a very low mean intensity of 0.525 (std = 0.333). These low means suggest that the robot's design and behaviors were not very effective at eliciting strong categorical emotional qualities, likely due to its non-humanoid, faceless design, which may limit its ability to convey these emotions convincingly. This bias towards lower perception of emotional intensities may also affect the predictive models, making it harder for them to accurately learn and predict these weak signals.

In contrast, the PAD dimensions do not show the same right-skewed pattern. The means for these dimensions are closer to the center of the scale, with pleasure at 4.525 (std = 1.229), arousal at 5.041 (std = 1.195), and dominance at 4.385 (std = 1.035). The higher mean for arousal suggests that certain behaviors, such as high wander speed, consistently elicited higher arousal responses. The substantial variability among participant responses for the PAD dimensions indicates that the robot's varied behaviors successfully elicited a wide range of responses, which is a positive outcome. This variability shows that while the emotional intensities were generally perceived as weak, the robot was still able to generate diverse emotional experiences in terms of PAD. This higher variability in the PAD dimensions provides the models with more data to learn from, potentially leading to better predictive accuracy.

These findings suggest that while the robot's behaviors could elicit a range of emotional responses, the strength of these responses for the basic emotions was generally weak. This implies that the base behaviors of the robot, as manipulated in this study, may not be inherently effective at conveying strong categorical emotional qualities. This consideration is crucial when interpreting the predictive models' performance, as they are based on data reflecting these relatively weak and varied emotional responses. Furthermore, the stronger variability and more centered responses in the PAD dimensions suggest better learning potential for predictive models in these areas.

Additionally, it is important to recognize that categorical emotions, which correspond to "universal" facial expressions, may not be abstract enough for a non-humanoid, faceless robot to effectively express. The robot's design inherently limits its ability to convey these discrete emotional states convincingly. PAD dimensions, being more abstract, might be inherently easier for participants to perceive in a robot that does not have a human-like appearance. This abstraction aligns better with the robot's capabilities. For instance, arousal is relatively straightforward to observe through behaviors such as high speed or intense movements. However, interpreting pleasure and dominance is more challenging. The concept of a robot being "dominant" is ambiguous and context-dependent, making it harder for participants to gauge accurately. This distinction suggests that while the robot can successfully convey levels of arousal, it struggles with more nuanced emotional cues like pleasure and dominance, reflecting a need for more intuitive design strategies to communicate these abstract qualities.

Moreover, the consistent overfitting observed in all random forest models and the underfitting seen in linear models for most dependent variables highlight a significant limitation of this study. This pattern is not unexpected given the nature of the data sampling method. The input parameters for the 512 videos were sampled using Sobol sequences, designed to uniformly cover the range of possible values for each parameter while minimizing correlation between samples. Consequently, dividing these points into training and testing datasets leaves a significant portion of the feature space unrepresented in each set, making it challenging for regression models to learn and predict the emotional qualities of the entire sample space effectively.

Consequently, when analyzing the predictive performance of these models, it's evident that the random forest models, despite their high performance on training data, tend to overfit, showing significantly better results on training data compared to testing data. This overfitting highlights their inability to generalize well to unseen data. On the other hand, linear models, which did not overfit, often underperformed on testing data, indicating that they lacked the complexity needed to capture the nuanced relationships between input parameters and emotional qualities.

For example, the results for pleasure intensity indicated modest improvements in predictive performance, with both linear models trained with X_{gmm} and X_{int} showing significant improvements over their baselines. However, the random forest model with X_{int} achieved the best accuracy, with an MSE of 1.308 and an R^2 of 0.127, suggesting some explanatory power but leaving much variance unexplained. Similarly, for arousal, the linear model trained with X_{gmm} achieved the best performance on unseen data, with an MSE of 0.785 and an R^2 of 0.429. However, the relatively low R^2 value indicates significant unexplained variance, suggesting a need for additional features or further refinement.

Ultimately, these findings emphasize the need for a more comprehensive evaluation method. To truly assess how well these models generalize, training them using the complete dataset of 512 points and testing their performance on new, unseen data collected through a new survey is essential. This approach would provide a more accurate assessment of the models' generalization capabilities, highlighting the limitations of the current dataset split and the need for extensive data collection to fully understand the predictive power and generalization ability of the regression models in capturing the emotional qualities elicited by the robot's behaviors.

5.3. Recommendations

Our study offers valuable insights into the connection between robotic behavioral parameters and perceived emotional qualities. However, the limitations we encountered emphasize the need for further research. Based on our findings, we suggest several directions for future research to deepen the understanding of emotion perception in human-robot interactions and enhance the robustness of predictive models.

Firstly, it is essential to test the generalizability of the models to new data. Collecting new data with different input parameters and participant groups would help validate the models' performance. This approach ensures that the models are not overfitting the current dataset and can reliably predict the emotional qualities of the robot's behaviors based on perceptions from a new group of participants. Additionally, a longitudinal study investigating how the emotional perception of the robot's behaviors evolves over time with repeated interactions between humans and the robot could provide deeper insights into the perception of emotions in appearance-constrained robots.

Secondly, examining whether our findings apply to other appearance-constrained robots is another crucial direction. While our study provides valuable insights into a specific appearance-constrained robot, testing these concepts across different robotic designs and interaction scenarios could help determine if the observed patterns are broadly applicable. For example, experimenting with different base behaviors for a robotic arm manipulator using modalities such as motion, light, and sound could yield new insights into designing emotional behaviors for appearance-constrained robots.

Lastly, expanding the sample size of both participants and videos would be beneficial to increase the feature and label spaces used for the training of regression models. A larger and more diverse participant pool could offer a more representative perspective on how various demographics perceive emotional qualities in robotic behaviors, thereby improving generalizability. Additionally, increasing the number of videos in the study would provide a broader data range, resulting in more nuanced and varied data in the dependent variables, which could ultimately enhance the robustness of the models.

5.4. Conclusion

In conclusion, this study demonstrates that variations in motion, light, and sound parameters significantly influence the perceived emotional qualities of an appearance-constrained robot's behaviors. Although the intensity of these emotional perceptions, particularly for basic emotion intensities, is relatively low, our findings indicate that manipulating factors such as speed, light temperature, and sound pitch results in discernible changes in emotional perception. For instance, higher speeds and pitches correlate with increased arousal, while warmer light colors affect both arousal and pleasure.

The regression modeling revealed notable limitations. Despite advanced feature engineering and modeling techniques yielding significant results in predicting emotional qualities, it is crucial to evaluate the generalizability of these models with new data. Our findings underscore the necessity for a comprehensive evaluation method. To accurately assess the models' generalization capabilities, they should be trained using the entire dataset of 512 points and tested on new, unseen data collected through a subsequent survey. This approach would provide a more precise evaluation of the models' ability to generalize and highlight areas for potential improvement.

Future research should examine whether these findings are applicable to other appearance-constrained robots. While this study offers valuable insights into a specific robot design, testing these concepts across various robotic designs and interaction scenarios could help determine if the observed patterns are broadly applicable. This could significantly contribute to the development of emotionally expressive robots in diverse contexts.

Moreover, the study's limitations include a relatively small feature space, resulting from the aggregation of data from 3160 points to 512 to mitigate the inherent noise in measuring emotional perception. Future research should focus on increasing the number of videos to generate a more representative sample of the feature space and recruiting more participants to ensure that the models can learn from a broader range of emotional perceptions, considering demographic diversity. Enhancing the dataset and participant pool would strengthen the robustness and applicability of the models developed in this study.

Despite these challenges, the study provides valuable insights into the different ways that motion, light, and sound parameters can influence the perception of the emotional qualities of an appearance-constrained robot. The findings highlight the potential for designing emotionally expressive behaviors for appearance-constrained robots. By building on this research, future work can focus on developing more effective and emotionally engaging robotic interactions, thereby enhancing user experiences and fostering more natural and intuitive human-robot relationships.

Bibliography

- [1] Francis M. Adams and Charles E. Osgood. A Cross-Cultural Study of the Affective Meanings of Color. *Journal of Cross-Cultural Psychology*, 4(2):135–156, June 1973. ISSN 0022-0221. doi: 10.1177/002202217300400201. URL <https://doi.org/10.1177/002202217300400201>. Publisher: SAGE Publications Inc.
- [2] Sami Alperen Akgun. Integrating Affective Expressions into Robot-Assisted Search and Rescue to Improve Human-Robot Communication. Master's thesis, University of Waterloo, 2021. URL <https://uwspace.uwaterloo.ca/handle/10012/17380>.
- [3] Sami Alperen Akgun, Moojan Ghafurian, Mark Crowley, and Kerstin Dautenhahn. Using Affect as a Communication Modality to Improve Human-Robot Communication in Robot-Assisted Search and Rescue Scenarios. *arXiv e-prints*, August 2022. doi: 10.48550/arXiv.2208.09580. URL <https://ui.adsabs.harvard.edu/abs/2022arXiv220809580A>. ADS Bibcode: 2022arXiv220809580A Type: article.
- [4] Lucy Anderson-Bashan, Benny Megidish, Hadas Erel, Iddo Wald, Guy Hoffman, Oren Zuckerman, and Andrey Grishko. The Greeting Machine: An Abstract Robotic Object for Opening Encounters. In *2018 27th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 595–602, August 2018. doi: 10.1109/ROMAN.2018.8525516. ISSN: 1944-9437.
- [5] Lisa Feldman Barrett, Ralph Adolphs, Stacy Marsella, Aleix M. Martinez, and Seth D. Pollak. Emotional Expressions Reconsidered: Challenges to Inferring Emotion From Human Facial Movements. *Psychological Science in the Public Interest*, 20(1):1–68, July 2019. ISSN 1529-1006. doi: 10.1177/1529100619832930. URL <https://doi.org/10.1177/1529100619832930>. Publisher: SAGE Publications Inc.
- [6] Paul H. Barrett. *The Works of Charles Darwin: Vol 23: The Expression of the Emotions in Man and Animals*. Routledge, London, June 2016. ISBN 978-1-315-47657-5. doi: 10.4324/9781315476575.
- [7] Christoph Bartneck, Dana Kulić, Elizabeth Croft, and Susana Zoghbi. Measurement Instruments for the Anthropomorphism, Animacy, Likeability, Perceived Intelligence, and Perceived Safety of Robots. *International Journal of Social Robotics*, 1(1):71–81, January 2009. ISSN 1875-4805. doi: 10.1007/s12369-008-0001-3. URL <https://doi.org/10.1007/s12369-008-0001-3>.
- [8] Alberto Betella, Martin Inderbitzin, Ulysses Bernardet, and Paul F.M.J. Verschure. Non-anthropomorphic Expression of Affective States through Parametrized Abstract Motifs. In *2013 Humaine Association Conference on Affective Computing and Intelligent Interaction*, pages 435–441, September 2013. doi: 10.1109/ACII.2013.78. ISSN: 2156-8111.
- [9] Cindy Bethel. *Robots Without Faces: Non-Verbal Social Human-Robot Interaction*. PhD Dissertation, University of South Florida, June 2009. URL <https://digitalcommons.usf.edu/etd/1855>.
- [10] Cindy L. Bethel and Robin R. Murphy. Affective expression in appearance constrained robots. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, HRI '06, pages 327–328, New York, NY, USA, March 2006. Association for Computing Machinery. ISBN 978-1-59593-294-5. doi: 10.1145/1121241.1121299. URL <https://doi.org/10.1145/1121241.1121299>.
- [11] Cindy L. Bethel and Robin R. Murphy. Survey of Non-facial/Non-verbal Affective Expressions for Appearance-Constrained Robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(1):83–92, January 2008. ISSN 1558-2442. doi: 10.1109/TSMCC.2007.905845. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews).
- [12] Cindy L. Bethel and Robin R. Murphy. Non-Facial and Non-Verbal Affective Expression for

- Appearance-Constrained Robots Used in Victim Management*. *Paladyn, Journal of Behavioral Robotics*, 1(4):219–230, December 2010. ISSN 2081-4836. doi: 10.2478/s13230-011-0009-5. URL <https://www.degruyter.com/document/doi/10.2478/s13230-011-0009-5/html?lang=en>. Publisher: De Gruyter Open Access.
- [13] Anthony J. Bishara and James B. Hittner. Testing the significance of a correlation with nonnormal data: comparison of Pearson, Spearman, transformation, and resampling approaches. *Psychological Methods*, 17(3):399–417, September 2012. ISSN 1939-1463. doi: 10.1037/a0028087.
- [14] Isabelle Blanchette and Anne Richards. The influence of affect on higher level cognition: A review of research on interpretation, judgement, decision making and reasoning. *Cognition and Emotion*, 24(4):561–595, June 2010. ISSN 0269-9931. doi: 10.1080/02699930903132496. URL <https://doi.org/10.1080/02699930903132496>. Publisher: Routledge _eprint: <https://doi.org/10.1080/02699930903132496>.
- [15] M. Bossema. *Effects of Robot Body Movements on the Adoption of the Intentional Stance*. PhD thesis, LIACS, Leiden University, 2020. URL <https://theses.liacs.nl/1946>.
- [16] Margaret M. Bradley and Peter J. Lang. Measuring emotion: The self-assessment manikin and the semantic differential. *Journal of Behavior Therapy and Experimental Psychiatry*, 25(1):49–59, March 1994. ISSN 0005-7916. doi: 10.1016/0005-7916(94)90063-9. URL <https://www.sciencedirect.com/science/article/pii/0005791694900639>.
- [17] Mason Bretan, Guy Hoffman, and Gil Weinberg. Emotionally expressive dynamic physical behaviors in robots. *International Journal of Human-Computer Studies*, 78:1–16, June 2015. ISSN 1071-5819. doi: 10.1016/j.ijhcs.2015.01.006. URL <https://www.sciencedirect.com/science/article/pii/S1071581915000191>.
- [18] Paul H. Bucci, X. Laura Cang, Hailey Mah, Laura Rodgers, and Karon E. MacLean. Real Emotions Don't Stand Still: Toward Ecologically Viable Representation of Affective Interaction. In *2019 8th International Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 1–7, September 2019. doi: 10.1109/ACII.2019.8925534. ISSN: 2156-8111.
- [19] Walter B. Cannon. The James-Lange Theory of Emotions: A Critical Examination and an Alternative Theory. *The American Journal of Psychology*, 39(1/4):106–124, 1927. ISSN 0002-9556. doi: 10.2307/1415404. URL <https://www.jstor.org/stable/1415404>. Publisher: University of Illinois Press.
- [20] Colleen M. Carpinella, Alisa B. Wyman, Michael A. Perez, and Steven J. Stroessner. The Robotic Social Attributes Scale (RoSAS): Development and Validation. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*, pages 254–262, Vienna Austria, March 2017. ACM. ISBN 978-1-4503-4336-7. doi: 10.1145/2909824.3020208. URL <https://dl.acm.org/doi/10.1145/2909824.3020208>.
- [21] Elizabeth Cha, Yunkyung Kim, Terrence Fong, and Maja J. Mataric. A Survey of Nonverbal Signaling Methods for Non-Humanoid Robots. *Foundations and Trends® in Robotics*, 6(4):211–323, February 2018. ISSN 1935-8253, 1935-8261. doi: 10.1561/23000000057. URL <https://www.nowpublishers.com/article/Details/ROB-057>. Publisher: Now Publishers, Inc.
- [22] Sonia Chernova and Andrea L. Thomaz. *Robot Learning from Human Teachers*. Morgan & Claypool Publishers, April 2014. ISBN 978-1-68173-179-7. Google-Books-ID: ulk7DwAAQBAJ.
- [23] Josep-Arnau Claret, Gentiane Venture, and Luis Basañez. Exploiting the Robot Kinematic Redundancy for Emotion Conveyance to Humans as a Lower Priority Task. *International Journal of Social Robotics*, 9(2):277–292, April 2017. ISSN 1875-4805. doi: 10.1007/s12369-016-0387-2. URL <https://doi.org/10.1007/s12369-016-0387-2>.
- [24] Alexandre Colle. The Role of Aesthetics in Robotics and the Rise of Polymorphic Robots. In *Companion of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, HRI '20, pages 623–624, New York, NY, USA, April 2020. Association for Computing Machinery. ISBN 978-1-4503-7057-8. doi: 10.1145/3371382.3379452. URL <https://dl.acm.org/doi/10.1145/3371382.3379452>.

- [25] Charles R Crowell, Jason C Deska, Michael Villano, Julaine Zenk, and John T Roddy Jr. Anthropomorphism of Robots: Study of Appearance and Agency. *JMIR Human Factors*, 6(2):e12629, May 2019. ISSN 2292-9495. doi: 10.2196/12629. URL <http://humanfactors.jmir.org/2019/2/e12629/>.
- [26] Shaundra B. Daily, Melva T. James, David Cherry, John J. Porter, Shelby S. Darnell, Joseph Isaac, and Tania Roy. Chapter 9 - Affective Computing: Historical Foundations, Current Applications, and Future Trends. In Myounghoon Jeon, editor, *Emotions and Affect in Human Factors and Human-Computer Interaction*, pages 213–231. Academic Press, San Diego, January 2017. ISBN 978-0-12-801851-4. doi: 10.1016/B978-0-12-801851-4.00009-4. URL <https://www.sciencedirect.com/science/article/pii/B9780128018514000094>.
- [27] Ting Dang, Vidhyasaharan Sethu, Julien Epps, and Eliathamby Ambikairajah. An Investigation of Emotion Prediction Uncertainty Using Gaussian Mixture Regression. In *Interspeech 2017*, pages 1248–1252. ISCA, August 2017. doi: 10.21437/Interspeech.2017-512. URL https://www.isca-archive.org/interspeech_2017/dang17_interspeech.html.
- [28] Ruta Desai, Fraser Anderson, Justin Matejka, Stelian Coros, James McCann, George Fitzmaurice, and Tovi Grossman. Geppetto: Enabling Semantic Design of Expressive Robot Behaviors. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, pages 1–14, New York, NY, USA, May 2019. Association for Computing Machinery. ISBN 978-1-4503-5970-2. doi: 10.1145/3290605.3300599. URL <https://doi.org/10.1145/3290605.3300599>.
- [29] Paul Ekman. An argument for basic emotions. *Cognition and Emotion*, 6(3-4):169–200, May 1992. ISSN 0269-9931, 1464-0600. doi: 10.1080/02699939208411068. URL <https://www.tandfonline.com/doi/full/10.1080/02699939208411068>.
- [30] Paul Ekman, E. Richard Sorenson, and Wallace V. Friesen. Pan-Cultural Elements in Facial Displays of Emotion. *Science*, 164(3875):86–88, April 1969. doi: 10.1126/science.164.3875.86. URL <https://www.science.org/doi/10.1126/science.164.3875.86>. Publisher: American Association for the Advancement of Science.
- [31] Frank Emmert-Streib and Matthias Dehmer. Evaluation of Regression Models: Model Assessment, Model Selection and Generalization Error. *Machine Learning and Knowledge Extraction*, 1(1): 521–551, March 2019. ISSN 2504-4990. doi: 10.3390/make1010032. URL <https://www.mdpi.com/2504-4990/1/1/32>. Number: 1 Publisher: Multidisciplinary Digital Publishing Institute.
- [32] Nicholas Epley, Adam Waytz, and John T. Cacioppo. On seeing human: A three-factor theory of anthropomorphism. *Psychological Review*, 114:864–886, 2007. ISSN 1939-1471. doi: 10.1037/0033-295X.114.4.864. Place: US Publisher: American Psychological Association.
- [33] Johannes Feldmaier, Tamara Marmat, Johannes Kuhn, and Klaus Diepold. Evaluation of a RGB-LED-based Emotion Display for Affective Agents, December 2016. URL <http://arxiv.org/abs/1612.07303>. arXiv:1612.07303 [cs].
- [34] Francesco Ferrari, Maria Paola Paladino, and Jolanda Jetten. Blurring Human–Machine Distinctions: Anthropomorphic Appearance in Social Robots as a Threat to Human Distinctiveness. *International Journal of Social Robotics*, 8(2):287–302, 2016. Publisher: Springer.
- [35] Emily Geisen. Using Commitment Requests Instead of Attention Checks, August 2022. URL <https://www.qualtrics.com/blog/attention-checks-and-data-quality/>.
- [36] Rachel Gockley, Jodi Forlizzi, and Reid Simmons. Interactions with a Moody Robot. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction*, pages 186–193, Salt Lake City Utah USA, March 2006. ACM. ISBN 978-1-59593-294-5. doi: 10.1145/1121241.1121274. URL <https://dl.acm.org/doi/10.1145/1121241.1121274>.
- [37] David Gouaillier, Vincent Hugel, Pierre Blazevic, Chris Kilner, Jerome Monceaux, Pascal Lafourcade, Brice Marnier, Julien Serre, and Bruno Maisonnier. Mechatronic design of NAO humanoid. In *2009 IEEE International Conference on Robotics and Automation*, pages 769–774, May 2009. doi: 10.1109/ROBOT.2009.5152516. ISSN: 1050-4729.
- [38] Mara Graziani, Lidia Dutkiewicz, Davide Calvaresi, José Pereira Amorim, Katerina Yordanova, Mor

- Vered, Rahul Nair, Pedro Henriques Abreu, Tobias Blanke, Valeria Pulignano, John O. Prior, Lode Lauwaert, Wessel Reijers, Adrien Depeursinge, Vincent Andrearczyk, and Henning Müller. A global taxonomy of interpretable AI: unifying the terminology for the technical and social sciences. *Artificial Intelligence Review*, September 2022. ISSN 1573-7462. doi: 10.1007/s10462-022-10256-8. URL <https://doi.org/10.1007/s10462-022-10256-8>.
- [39] James J. Gross and Lisa Feldman Barrett. Emotion Generation and Emotion Regulation: One or Two Depends on Your Point of View. *Emotion Review*, 3(1):8–16, January 2011. ISSN 1754-0739. doi: 10.1177/1754073910380974. URL <https://doi.org/10.1177/1754073910380974>. Publisher: SAGE Publications.
- [40] Erico Guizzo. By leaps and bounds: An exclusive look at how Boston dynamics is redefining robot agility. *IEEE Spectrum*, 56(12):34–39, December 2019. ISSN 1939-9340. doi: 10.1109/MSPEC.2019.8913831. URL https://ieeexplore.ieee.org/abstract/document/8913831?casa_token=GWxp8WDVf9AAAAAA:xaIHXpc2SCX1r4hJzD-8Wwm9X93iXEFCG0F75QyZcrToq5ptkoPyZ8n53D8fDiQpJsOUzx4. Conference Name: IEEE Spectrum.
- [41] Guy Hoffman and Wendy Ju. Designing robots with movement in mind. *Journal of Human-Robot Interaction*, 3(1):91–122, February 2014. doi: 10.5898/JHRI.3.1.Hoffman. URL <https://dl.acm.org/doi/10.5898/JHRI.3.1.Hoffman>.
- [42] Guy Hoffman, Oren Zuckerman, Gilad Hirschberger, Michal Luria, and Tal Shani Sherman. Design and Evaluation of a Peripheral Robotic Conversation Companion. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction, HRI '15*, pages 3–10, New York, NY, USA, March 2015. Association for Computing Machinery. ISBN 978-1-4503-2883-8. doi: 10.1145/2696454.2696495. URL <https://dl.acm.org/doi/10.1145/2696454.2696495>.
- [43] Marius Hoggenmueller, Jiahao Chen, and Luke Hespanhol. Emotional expressions of non-humanoid urban robots: the role of contextual aspects on interpretations. In *Proceedings of the 9TH ACM International Symposium on Pervasive Displays, PerDis '20*, pages 87–95, New York, NY, USA, June 2020. Association for Computing Machinery. ISBN 978-1-4503-7986-1. doi: 10.1145/3393712.3395341. URL <https://doi.org/10.1145/3393712.3395341>.
- [44] Rachael E. Jack, Oliver G. B. Garrod, Hui Yu, Roberto Caldara, and Philippe G. Schyns. Facial expressions of emotion are not culturally universal. *Proceedings of the National Academy of Sciences*, 109(19):7241–7244, May 2012. doi: 10.1073/pnas.1200155109. URL <https://www.pnas.org/doi/abs/10.1073/pnas.1200155109>. Publisher: Proceedings of the National Academy of Sciences.
- [45] Eunju Jeong, Gyu Hyun Kwon, and Junseop So. Exploring the taxonomic and associative link between emotion and function for robot sound design. In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 641–643, June 2017. doi: 10.1109/URAI.2017.7992692.
- [46] Keunwook Kim, Jaeyeon Park, and Taeyun Li. Post-plant: A Series of Non-humanoid Robots with Embedded Physical Non-verbal Interaction: The Development of Non-Verbal Human-Robot Interaction Framework and Input/Output Integrated Motor Interface. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems, CHI EA '21*, pages 1–3, New York, NY, USA, May 2021. Association for Computing Machinery. ISBN 978-1-4503-8095-9. doi: 10.1145/3411763.3451560. URL <https://dl.acm.org/doi/10.1145/3411763.3451560>.
- [47] Troels Aske Klausen, Ulrich Farhadi, Evgenios Vlachos, and Jonas Jørgensen. Signalling Emotions with a Breathing Soft Robot. In *2022 IEEE 5th International Conference on Soft Robotics (RoboSoft)*, pages 194–200, April 2022. doi: 10.1109/RoboSoft54090.2022.9762140.
- [48] Jens Kober, J. Andrew Bagnell, and Jan Peters. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, September 2013. ISSN 0278-3649, 1741-3176. doi: 10.1177/0278364913495721. URL <http://journals.sagepub.com/doi/10.1177/0278364913495721>.
- [49] Takanori Komatsu. Toward Making Humans Empathize with Artificial Agents by Means of Subtle

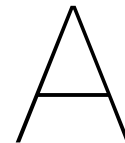
- Expressions. In Jianhua Tao, Tieniu Tan, and Rosalind W. Picard, editors, *Affective Computing and Intelligent Interaction*, Lecture Notes in Computer Science, pages 458–465, Berlin, Heidelberg, 2005. Springer. ISBN 978-3-540-32273-3. doi: 10.1007/11573548_59.
- [50] Anagha Kulkarni, Sarath Sreedharan, Sarah Keren, Tathagata Chakraborti, David E. Smith, and Subbarao Kambhampati. Designing Environments Conducive to Interpretable Robot Behavior. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10982–10989, October 2020. doi: 10.1109/IROS45743.2020.9340832. ISSN: 2153-0866.
- [51] Jory Lafaye, David Gouaillier, and Pierre-Brice Wieber. Linear model predictive control of the locomotion of Pepper, a humanoid robot with omnidirectional wheels. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 336–341, November 2014. doi: 10.1109/HUMANOIDS.2014.7041381. URL https://ieeexplore.ieee.org/abstract/document/7041381?casa_token=5bnSHqX7nUYAAAAA:neRsLfD2ZPAC51YuoKt3Krb3-AY7IBKiXbBlIjdjBWUG0wcWM-7Hy8EJj9tz8ha_4Rmmhv0. ISSN: 2164-0580.
- [52] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, May 2015. ISSN 1476-4687. doi: 10.1038/nature14539. URL <https://www.nature.com/articles/nature14539>. Number: 7553 Publisher: Nature Publishing Group.
- [53] Diana Löffler, Nina Schmidt, and Robert Tscharn. Multimodal Expression of Artificial Emotion in Social Robots Using Color, Motion and Sound. In *2018 13th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 334–343, March 2018. ISSN: 2167-2148.
- [54] Marco Matarese, Alessandra Sciutti, Francesco Rea, and Silvia Rossi. Toward Robots’ Behavioral Transparency of Temporal Difference Reinforcement Learning With a Human Teacher. *IEEE Transactions on Human-Machine Systems*, 51(6):578–589, December 2021. ISSN 2168-2305. doi: 10.1109/THMS.2021.3116119. Conference Name: IEEE Transactions on Human-Machine Systems.
- [55] Andrii Matviienko, Andreas Löcken, Abdallah El Ali, Wilko Heuten, and Susanne Boll. NaviLight: investigating ambient light displays for turn-by-turn navigation in cars. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services, MobileHCI ’16*, pages 283–294, New York, NY, USA, September 2016. Association for Computing Machinery. ISBN 978-1-4503-4408-1. doi: 10.1145/2935334.2935359. URL <https://dl.acm.org/doi/10.1145/2935334.2935359>.
- [56] Albert Mehrabian. Pleasure-arousal-dominance: A general framework for describing and measuring individual differences in Temperament. *Current Psychology*, 14(4):261–292, December 1996. ISSN 1936-4733. doi: 10.1007/BF02686918. URL <https://doi.org/10.1007/BF02686918>.
- [57] Albert Mehrabian and James A. Russell. The Basic Emotional Impact of Environments. *Perceptual and Motor Skills*, 38(1):283–301, February 1974. ISSN 0031-5125. doi: 10.2466/pms.1974.38.1.283. URL <https://doi.org/10.2466/pms.1974.38.1.283>. Publisher: SAGE Publications Inc.
- [58] Giorgio Metta, Lorenzo Natale, Francesco Nori, Giulio Sandini, David Vernon, Luciano Fadiga, Claes von Hofsten, Kerstin Rosander, Manuel Lopes, José Santos-Victor, Alexandre Bernardino, and Luis Montesano. The iCub humanoid robot: An open-systems platform for research in cognitive development. *Neural Networks*, 23(8):1125–1134, October 2010. ISSN 0893-6080. doi: 10.1016/j.neunet.2010.08.010. URL <https://www.sciencedirect.com/science/article/pii/S0893608010001619>.
- [59] Marvin Minsky. *Society Of Mind*. Simon and Schuster, March 1988. ISBN 978-0-671-65713-0. Google-Books-ID: bLDLlIfRpdKc.
- [60] Alban Nanty and Rodolphe Gelin. Fuzzy Controlled PAD Emotional State of a NAO Robot. In *2013 Conference on Technologies and Applications of Artificial Intelligence*, pages 90–96, December 2013. doi: 10.1109/TAAI.2013.30. ISSN: 2376-6824.
- [61] Jekaterina Novikova, Gang Ren, and Leon Watts. It’s Not the Way You Look, It’s How You Move: Validating a General Scheme for Robot Affective Behaviour. In Julio Abascal, Simone Barbosa, Mirko Fetter, Tom Gross, Philippe Palanque, and Marco Winckler, editors, *Human-Computer*

- Interaction – INTERACT 2015*, Lecture Notes in Computer Science, pages 239–258, Cham, 2015. Springer International Publishing. ISBN 978-3-319-22698-9. doi: 10.1007/978-3-319-22698-9_16.
- [62] Art B. Owen. On dropping the first Sobol' point, December 2021. URL <http://arxiv.org/abs/2008.08051>. arXiv:2008.08051 [cs, math, stat].
- [63] Andrea Papenmeier. *Affect Expression through Robot Motion in Minimalistic Robots*. Research Topics, University of Twente, 2018.
- [64] Marina Pavlova, Arseny A Sokolov, and Alexander Sokolov. Perceived Dynamics of Static Images Enables Emotional Attribution. *Perception*, 34(9):1107–1116, September 2005. ISSN 0301-0066. doi: 10.1068/p5400. URL <https://doi.org/10.1068/p5400>. Publisher: SAGE Publications Ltd STM.
- [65] Robert Plutchik and Henry Kellerman, editors. *Emotion: Theory, Research, and Experience*. Academic Press, New York, 1980. ISBN 978-0-12-558701-3.
- [66] Daniel J. Rea, James E. Young, and Pourang Irani. The Roomba mood ring: an ambient-display robot. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, HRI '12, pages 217–218, New York, NY, USA, March 2012. Association for Computing Machinery. ISBN 978-1-4503-1063-5. doi: 10.1145/2157689.2157763. URL <https://doi.org/10.1145/2157689.2157763>.
- [67] Thomas Riccio. Sophia Robot: An Emergent Ethnography. *TDR*, 65(3):42–77, September 2021. ISSN 1054-2043, 1531-4715. doi: 10.1017/S1054204321000319. URL <https://www.cambridge.org/core/journals/the-drama-review/article/sophia-robot/724ECA22CFC406398D70DE0443B2FF0F>. Publisher: Cambridge University Press.
- [68] Jaime A. Rincon, Angelo Costa, Paulo Novais, Vicente Julian, and Carlos Carrascosa. A new emotional robot assistant that facilitates human interaction and persuasion. *Knowledge and Information Systems*, 60(1):363–383, July 2019. ISSN 0219-3116. doi: 10.1007/s10115-018-1231-9. URL <https://doi.org/10.1007/s10115-018-1231-9>.
- [69] Liz Rincon, Enrique Coronado, Hansen Hendra, Julyando Phan, Zur Zainalkefli, and Gentiane Venture. Expressive states with a robot arm using adaptive fuzzy and robust predictive controllers. In *2018 3rd International Conference on Control and Robotics Engineering (ICCRE)*, pages 11–15, April 2018. doi: 10.1109/ICCRE.2018.8376425.
- [70] Fabien Ringeval, Andreas Sonderegger, Juergen Sauer, and Denis Lalanne. Introducing the RECOLA multimodal corpus of remote collaborative and affective interactions. In *2013 10th IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, pages 1–8, April 2013. doi: 10.1109/FG.2013.6553805. URL https://ieeexplore.ieee.org/abstract/document/6553805?casa_token=8zXhjmFtbeAAAAA:alkDwMAQEnOkVrcDak8YKnC5Z3ipYzehkJTaAVcfz0ABDQM5LZboMWI4UYemon13TKtG4RE.
- [71] Alwin de Rooij, Joost Broekens, and Maarten H. Lamers. Abstract Expressions of Affect. *International Journal of Synthetic Emotions (IJSE)*, 4(1):1–31, January 2013. ISSN 1947-9093. doi: 10.4018/jse.2013010101. URL <https://www.igi-global.com/article/content/www.igi-global.com/article/content/77654>. Publisher: IGI Global.
- [72] James A Russell. A Circumplex Model of Affect. *Journal of Personality and Social Psychology*, 39(6):1161, 1980. Publisher: American Psychological Association.
- [73] James A Russell and Albert Mehrabian. Evidence for a three-factor theory of emotions. *Journal of Research in Personality*, 11(3):273–294, September 1977. ISSN 0092-6566. doi: 10.1016/0092-6566(77)90037-X. URL <https://www.sciencedirect.com/science/article/pii/009265667790037X>.
- [74] Martin Saerbeck and Christoph Bartneck. Perception of affect elicited by robot motion. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 53–60, March 2010. doi: 10.1109/HRI.2010.5453269. ISSN: 2167-2148.
- [75] Elaheh Sanoubari, Byron David, Chase Kew, Corbin Cunningham, and Ken Caluwaerts. From

Message to Expression: Exploring Non-Verbal Communication for Appearance-Constrained Robots. In *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1193–1200, August 2022. doi: 10.1109/RO-MAN53752.2022.9900520. ISSN: 1944-9437.

- [76] Shane Saunderson and Goldie Nejat. How Robots Influence Humans: A Survey of Nonverbal Communication in Social Human–Robot Interaction. *International Journal of Social Robotics*, 11(4):575–608, August 2019. ISSN 1875-4805. doi: 10.1007/s12369-019-00523-0. URL <https://doi.org/10.1007/s12369-019-00523-0>.
- [77] Richard Savery and Gil Weinberg. A Survey of Robotics and Emotion: Classifications and Models of Emotional Interaction. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 986–993, August 2020. doi: 10.1109/RO-MAN47096.2020.9223536. ISSN: 1944-9437.
- [78] Markus Schwenk and Kai O. Arras. R2-D2 Reloaded: A flexible sound synthesis system for sonic human-robot interaction design. In *The 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 161–167, August 2014. doi: 10.1109/ROMAN.2014.6926247. ISSN: 1944-9437.
- [79] Philip Sedgwick. Multiple significance tests: the Bonferroni correction. *BMJ*, 344:e509, January 2012. ISSN 0959-8138, 1468-5833. doi: 10.1136/bmj.e509. URL <https://www.bmj.com/content/344/bmj.e509>. Publisher: British Medical Journal Publishing Group Section: Endgames.
- [80] Daniel B. Shank, Christopher Graves, Alexander Gott, Patrick Gamez, and Sophia Rodriguez. Feeling our way to machine minds: People’s emotions when perceiving mind in artificial intelligence. *Computers in Human Behavior*, 98:256–266, September 2019. ISSN 0747-5632. doi: 10.1016/j.chb.2019.04.001. URL <https://www.sciencedirect.com/science/article/pii/S0747563219301396>.
- [81] Satoshi Shigemi. ASIMO and Humanoid Robot Research at Honda. In Ambarish Goswami and Prahlad Vadakkepat, editors, *Humanoid Robotics: A Reference*, pages 55–90. Springer Netherlands, Dordrecht, 2019. ISBN 978-94-007-6045-5 978-94-007-6046-2. doi: 10.1007/978-94-007-6046-2_9. URL http://link.springer.com/10.1007/978-94-007-6046-2_9.
- [82] Keng Siau and Weiyu Wang. Building Trust in Artificial Intelligence, Machine Learning, and Robotics. *Cutter Business Technology Journal*, 31(2):47–53, 2018.
- [83] Bruno Siciliano and Oussama Khatib. Humanoid Robots: Historical Perspective, Overview, and Scope. In Ambarish Goswami and Prahlad Vadakkepat, editors, *Humanoid Robotics: A Reference*, pages 3–8. Springer Netherlands, Dordrecht, 2019. ISBN 978-94-007-6046-2. doi: 10.1007/978-94-007-6046-2_64. URL https://doi.org/10.1007/978-94-007-6046-2_64.
- [84] I. M Sobol’. On the distribution of points in a cube and the approximate evaluation of integrals. *USSR Computational Mathematics and Mathematical Physics*, 7(4):86–112, January 1967. ISSN 0041-5553. doi: 10.1016/0041-5553(67)90144-9. URL <https://www.sciencedirect.com/science/article/pii/0041555367901449>.
- [85] Sichao Song and Seiji Yamada. Expressing Emotions through Color, Sound, and Vibration with an Appearance-Constrained Social Robot. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction, HRI ’17*, pages 2–11, New York, NY, USA, March 2017. Association for Computing Machinery. ISBN 978-1-4503-4336-7. doi: 10.1145/2909824.3020239. URL <https://doi.org/10.1145/2909824.3020239>.
- [86] Sichao Song and Seiji Yamada. Designing Expressive Lights and In-Situ Motions for Robots to Express Emotions. In *Proceedings of the 6th International Conference on Human-Agent Interaction*, pages 222–228, Southampton United Kingdom, December 2018. ACM. ISBN 978-1-4503-5953-5. doi: 10.1145/3284432.3284458. URL <https://dl.acm.org/doi/10.1145/3284432.3284458>.
- [87] Sichao Song and Seiji Yamada. Designing LED lights for a robot to communicate gaze. *Advanced Robotics*, 33(7-8):360–368, April 2019. ISSN 0169-1864. doi: 10.1080/01691864.2019.1600426. URL <https://doi.org/10.1080/01691864.2019.1600426>. Publisher: Taylor & Francis _eprint: <https://doi.org/10.1080/01691864.2019.1600426>.

- [88] S. Sotnik and V. Lyashenko. Overview of Innovative Walking Robots. *International Journal of Academic Engineering Research (IJAER)*, 6:3–7, 2022. ISSN 2643-9085. URL <https://openarchive.nure.ua/handle/document/20233>.
- [89] Paul Strohmeier, Juan Pablo Carrascal, Bernard Cheng, Margaret Meban, and Roel Vertegaal. An Evaluation of Shape Changes for Conveying Emotions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 3781–3792, New York, NY, USA, May 2016. Association for Computing Machinery. ISBN 978-1-4503-3362-7. doi: 10.1145/2858036.2858537. URL <https://dl.acm.org/doi/10.1145/2858036.2858537>.
- [90] Howard E. Sypher, Beverly Davenport Sypher, and John W. Haas. Getting Emotional: The Role of Affect in Interpersonal Communication. *American Behavioral Scientist*, 31(3):372–383, January 1988. ISSN 0002-7642. doi: 10.1177/000276488031003008. URL <https://doi.org/10.1177/000276488031003008>. Publisher: SAGE Publications Inc.
- [91] Haodan Tan, Liping Sun, and Selma Šabanović. Feeling green: Empathy affects perceptions of usefulness and intention to use a robotic recycling bin. In *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*, pages 1051–1056, August 2016. doi: 10.1109/ROMAN.2016.7745238. ISSN: 1944-9437.
- [92] Haodan Tan, John Tiab, Selma Šabanović, and Kasper Hornbæk. Happy Moves, Sad Grooves: Using Theories of Biological Motion and Affect to Design Shape-Changing Interfaces. In *Proceedings of the 2016 ACM Conference on Designing Interactive Systems*, DIS '16, pages 1282–1293, New York, NY, USA, June 2016. Association for Computing Machinery. ISBN 978-1-4503-4031-1. doi: 10.1145/2901790.2901845. URL <https://dl.acm.org/doi/10.1145/2901790.2901845>.
- [93] Amirali Soltani Tehrani, Niloufar Faridani, and Ramin Toosi. Unsupervised Representations Improve Supervised Learning in Speech Emotion Recognition, September 2023. URL <http://arxiv.org/abs/2309.12714>. arXiv:2309.12714 [cs, eess].
- [94] C. Trevarthen. Facial expressions of emotion in mother-infant interaction. *Human neurobiology*, 4(1):21–32, January 1985. ISSN 0721-9075.
- [95] Evgenios Vlachos and Henrik Schärfe. Towards Designing Android Faces After Actual Humans. In Gordan Jezic, Robert J. Howlett, and Lakhmi C. Jain, editors, *Agent and Multi-Agent Systems: Technologies and Applications*, Smart Innovation, Systems and Technologies, pages 109–119, Cham, 2015. Springer International Publishing. ISBN 978-3-319-19728-9. doi: 10.1007/978-3-319-19728-9_9.
- [96] Steve Whittaker, Yvonne Rogers, Elena Petrovskaya, and Hongbin Zhuang. Designing Personas for Expressive Robots: Personality in the New Breed of Moving, Speaking, and Colorful Social Home Robots. *ACM Transactions on Human-Robot Interaction*, 10(1):8:1–8:25, February 2021. doi: 10.1145/3424153. URL <https://dl.acm.org/doi/10.1145/3424153>.
- [97] Patrick Henry Winston. *Artificial intelligence (2nd ed.)*. Addison-Wesley Longman Publishing Co., Inc., USA, 1984. ISBN 978-0-201-08259-3.
- [98] Robert H. Wortham and Andreas Theodorou. Robot transparency, trust and utility. *Connection Science*, 29(3):242–248, July 2017. ISSN 0954-0091. doi: 10.1080/09540091.2017.1313816. URL <https://doi.org/10.1080/09540091.2017.1313816>. Publisher: Taylor & Francis _eprint: <https://doi.org/10.1080/09540091.2017.1313816>.
- [99] Yangwei Ying, Yuanwu Tu, and Hong Zhou. Unsupervised Feature Learning for Speech Emotion Recognition Based on Autoencoder. *Electronics*, 10(17):2086, January 2021. ISSN 2079-9292. doi: 10.3390/electronics10172086. URL <https://www.mdpi.com/2079-9292/10/17/2086>. Number: 17 Publisher: Multidisciplinary Digital Publishing Institute.
- [100] Anna D Zych and Nadine Gogolla. Expressions of emotions across species. *Current Opinion in Neurobiology*, 68:57–66, June 2021. ISSN 0959-4388. doi: 10.1016/j.conb.2021.01.003. URL <https://www.sciencedirect.com/science/article/pii/S0959438821000052>.



Code

A.1. Wander Base Behavior

```
#include <Arduino.h> // Include the main Arduino library for basic functions and macros
#include <MeMCore.h> // Include the Makeblock library for controlling Makeblock components
#include <SoftwareSerial.h> // Include the library for serial communication on digital
    pins
#include <Wire.h> // Include the library for I2C communication
#include <stdlib.h> // Include standard library for general purpose functions
#include <math.h> // Include math library for mathematical operations

// Timer control variables for managing time-based actions
double currentTime = 0; // Stores the current time in seconds
double lastTime = 0; // Stores the last time an action was taken in seconds

// Initialize components attached to the robot
MeLineFollower linefollower_2(2); // Line follower module on port 2
MeDCMotor motor_9(9); // DC motor connected to port 9
MeDCMotor motor_10(10); // DC motor connected to port 10

// Variables to control the wander behavior
double turnDuration; // Duration of the turning action in seconds
double forwardDuration; // Duration of the forward movement in seconds
double lineTurnDuration; // Duration of the turn when a line is detected
double targetForwardSpeed; // Desired speed for forward movement
double targetTurnSpeed; // Desired speed for turning
int acceleration; // Factor to increase/decrease speed gradually
int wanderCycle; // Counter for the number of wander cycles completed

// Flag to determine if the inputs for wander behavior are valid
boolean doWander;

// General input parameters for the wander behavior
double duration = 10; // Total duration of the wander behavior in seconds
boolean stayInBounds = true; // Flag to stay within a bounded area

// Specific input parameters for configuring the wander behavior
double wanderSpeed = 100; // Base speed for wandering
double wanderSlope = 0; // Slope for changing speed dynamically
double wanderRoundness = 0.5; // Factor for adjusting the sharpness of turns
double wanderTurnToForwardRatio = 0.9; // Ratio of turn duration to forward duration
    within a cycle
double wanderCycleRate = 2; // Number of cycles per second
double wanderCycleStandardDeviation = 0.5; // Variability in the cycle rate
double wanderSpeedStandardDeviation = 0.5; // Variability in the speed
double wanderPhase = 0; // Initial phase delay before starting to wander

// Function to calculate the elapsed time since the last reset
double getLastTime()
{
    // Update currentTime, calculate and return the elapsed time since last reset
    return currentTime = millis() / 1000.0 - lastTime;
}

// Custom delay function to pause execution for a given number of seconds
```

```

void _delay(double seconds)
{
    long endTime = millis() + seconds * 1000; // Calculate end time in milliseconds
    while (millis() < endTime) _loop(); // Loop until the end time is reached
}

// Function to control the robot's movement in a specified direction at a specified speed
void move(int direction, int speed)
{
    int leftSpeed = 0; // Speed of the left motor
    int rightSpeed = 0; // Speed of the right motor

    // Determine the speed of each motor based on the desired direction of movement
    if (direction == 1) {
        // Forward
        leftSpeed = speed;
        rightSpeed = speed;
    } else if (direction == 2) {
        // Backward
        leftSpeed = -speed;
        rightSpeed = -speed;
    } else if (direction == 3) {
        // Left
        leftSpeed = -speed;
        rightSpeed = speed;
    } else if (direction == 4) {
        // Right
        leftSpeed = speed;
        rightSpeed = -speed;
    }

    // Apply the calculated speeds to the motors
    motor_9.run(9 == M1 ? -leftSpeed : leftSpeed); // Adjust direction based on motor
    // configuration
    motor_10.run(10 == M1 ? -rightSpeed : rightSpeed);
}

// Generates a random number with a Gaussian distribution around a mean of 0
double GenerateGaussian(double standardDeviation)
{
    double u1 = rand() / (RAND_MAX + 1.0);
    double u2 = rand() / (RAND_MAX + 1.0);
    double z0 = sqrt(-2.0 * log(u1)) * cos(2.0 * M_PI * u2);
    return z0 * standardDeviation;
}

// Generates a random number within a specified range [min, max]
double GetRandomNumber(double min, double max)
{
    double randomNumber = (double)rand() / RAND_MAX; // Generate a random value between 0
    // and 1
    return (randomNumber * (max - min)) + min; // Scale and shift to the specified range
}

// Caps a number within a specified range [lowerLimit, upperLimit]
void CapNumber(double* number, double lowerLimit, double upperLimit)
{
    if (*number < lowerLimit) {
        *number = lowerLimit; // Cap to lower limit
    }
    if (*number > upperLimit) {
        *number = upperLimit; // Cap to upper limit
    }
}

// Checks whether the input parameters for wander behavior are within valid ranges
void CheckValidWanderInput(double speed, double slope, double roundness, double
    turnToForwardRatio, double cycleRate, double cycleStandardDeviation, double
    speedStandardDeviation, double phase)
{
    // Validate each input parameter against its acceptable range
    boolean isValidSpeed = speed >= 25 && speed <= 100;
    boolean isValidSlope = slope >= -5 && slope <= 5;
}

```

```

boolean isValidRoundness = roundness >= 0 && roundness <= 1;
boolean isValidTurnToForwardRatio = turnToForwardRatio <= 1 && turnToForwardRatio > 0;
boolean isCycleRatePositive = cycleRate > 0;
boolean isValidStandardDeviation = cycleStandardDeviation >= 0 && speedStandardDeviation
    >= 0;
boolean isValidPhase = phase >= 0;

// Set doWander to true only if all parameters are valid
doWander = isValidSpeed && isValidSlope && isValidRoundness && isValidTurnToForwardRatio
    && isCycleRatePositive && isValidStandardDeviation && isValidPhase;
}

// Sets the durations for forward movement, turning, and line avoidance turning
void SetWanderDurations(double speed, double turnToForwardRatio, double cycleRate, double
    cycleStandardDeviation)
{
    forwardDuration = (1 - turnToForwardRatio) / cycleRate; // Calculate base forward
        duration
    forwardDuration += GenerateGaussian(cycleStandardDeviation); // Add variability
    CapNumber(&forwardDuration, 0, 1 / cycleRate); // Ensure duration is within bounds

    turnDuration = 1 / cycleRate - forwardDuration; // Calculate turn duration based on
        remaining time in the cycle

    // Calculate line turn duration based on speed
    if (speed <= 100 && speed >= 50) {
        lineTurnDuration = 0.5;
    } else {
        lineTurnDuration = 0.0008 * pow(speed, 2) - 0.11 * speed + 4;
    }
}

// Sets the target speeds for forward movement and turning
void SetTargetSpeeds(double speed, double roundness, double speedStandardDeviation)
{
    targetForwardSpeed = speed + GenerateGaussian(speedStandardDeviation); // Add
        variability to forward speed
    CapNumber(&targetForwardSpeed, 25, 100); // Ensure speed is within bounds

    // Calculate target turn speed based on forward speed and roundness
    targetTurnSpeed = -targetForwardSpeed / 2 + 4 * targetForwardSpeed * roundness - 4 *
        targetForwardSpeed * pow(roundness, 2);
}

// Method to execute the wander base behavior
void Wander(double duration, boolean stayInBounds,
    double wanderSpeed, double wanderSlope, double wanderRoundness, double
    wanderTurnToForwardRatio, double wanderCycleRate, double wanderCycleStandardDeviation,
    double wanderSpeedStandardDeviation, double wanderPhase)
{
    // Validates the input parameters for the wander behavior.
    CheckValidWanderInput(wanderSpeed, wanderSlope, wanderRoundness,
        wanderTurnToForwardRatio, wanderCycleRate, wanderCycleStandardDeviation,
        wanderSpeedStandardDeviation, wanderPhase);

    // Proceeds with the wander behavior if the inputs are validated successfully.
    if (doWander) {
        // Calculates and sets target speeds for both forward movement and turning based on
            inputs and variability.
        SetTargetSpeeds(wanderSpeed, wanderRoundness, wanderSpeedStandardDeviation);

        // Determines durations for forward movement, turning, and line-avoidance based on
            inputs and variability.
        SetWanderDurations(wanderSpeed, wanderTurnToForwardRatio, wanderCycleRate,
            wanderCycleStandardDeviation);

        // Initializes the counter to keep track of completed wander cycles.
        wanderCycle = 0;

        // Sets the initial acceleration for speed adjustment during forward movement, used
            when slope != 0.
        acceleration = 1;
    }
}

```



```

// Records the start time of the wander behavior.
lastTime = millis() / 1000.0;

// Continues executing the wander behavior for the specified duration.
while (!(getLastTime() > duration)) {
  _loop();

  boolean isTurning = false;

  // If valid wander input is given
  if (doWander) {
    // Start executing after the given phase
    if (getLastTime() - wanderPhase >= 0) {
      // Turn around if a black line is detected
      if (stayInBounds) {
        // Get sensor reading
        int sensorReading = linefollower_2.readSensors();

        // Turn left if right sensor detects a white line
        if (!isTurning && (0 ? (1 == 0 ? sensorReading == 0 : (sensorReading & 1) == 1)
          : (1 == 0 ? sensorReading == 3 : (sensorReading & 1) == 0))) {
          isTurning = true;
          move(3, targetForwardSpeed / 100.0 * 255);
          _delay(lineTurnDuration);
          move(3, 0);
        }

        // Turn right if left sensor detects a white line
        if (!isTurning && (0 ? (2 == 0 ? sensorReading == 0 : (sensorReading & 2) == 2)
          : (2 == 0 ? sensorReading == 3 : (sensorReading & 2) == 0))) {
          isTurning = true;
          move(4, targetForwardSpeed / 100.0 * 255);
          _delay(lineTurnDuration);
          move(4, 0);
        }
      }
    }
  }

  // Determines if it's time to move forward based on the current phase and cycle rate
  if (wanderSlope == 0) {
    // Moves forward at a constant speed if there's no slope.
    if (getLastTime() - wanderPhase < wanderCycle / wanderCycleRate + forwardDuration)
    {
      move(1, targetForwardSpeed / 100.0 * 255);
    }
  } else if (wanderSlope > 0) {
    // Gradually increases speed if the slope is positive.
    if (getLastTime() - wanderPhase < wanderCycle / wanderCycleRate + acceleration *
forwardDuration / 100) {
      move(1, wanderSlope * acceleration * targetForwardSpeed / 100 / 100.0 * 255);
      if (acceleration < 100) {
        acceleration += 1; // Increment acceleration until it reaches 100.
      }
    }
  } else if (wanderSlope < 0) {
    // Gradually decreases speed if the slope is negative.
    if (getLastTime() - wanderPhase < wanderCycle / wanderCycleRate + acceleration *
forwardDuration / 100) {
      move(1, wanderSlope * (acceleration - 100) * targetForwardSpeed / 100 / 100.0 *
255);
      if (acceleration < 100) {
        acceleration += 1; // Increment acceleration until it reaches 100, for
decreasing speed.
      }
    }
  }

  // Initiates a turn after the forward movement phase is complete.
  if (getLastTime() - wanderPhase > wanderCycle / wanderCycleRate + forwardDuration &&
    getLastTime() - wanderPhase < (wanderCycle + 1) / wanderCycleRate) {
    // Alternates turning direction with each cycle.

```

```

    if (fmod(wanderCycle, 2) == 0) {
        // Executes a right turn on even cycles.
        motor_9.run(-targetForwardSpeed / 100.0 * 255);
        motor_10.run(targetTurnSpeed / 100.0 * 255);
    } else {
        // Executes a left turn on odd cycles.
        motor_9.run(-targetTurnSpeed / 100.0 * 255);
        motor_10.run(targetForwardSpeed / 100.0 * 255);
    }
}

// Prepares for the next cycle after completing both forward movement and turning.
if (getLastTime() - wanderPhase > (wanderCycle + 1) / wanderCycleRate) {
    wanderCycle += 1; // Increments the cycle counter.
    acceleration = 1; // Resets acceleration for the next cycle.

    // Recalculates speeds and durations for the next cycle, incorporating variability
    .
    SetTargetSpeeds(wanderSpeed, wanderRoundness, wanderSpeedStandardDeviation);
    SetWanderDurations(wanderSpeed, wanderTurnToForwardRatio, wanderCycleRate,
wanderCycleStandardDeviation);
}
}

// Stops the motors to halt the robot's movement at the end of the wander behavior.
motor_9.run(0);
motor_10.run(0);
}

// Setup function to initialize the robot and start the wander behavior
void setup()
{
    randomSeed(0); // Initialize the random number generator seed.

    // Start the wandering behavior with the specified parameters.
    Wander(duration, stayInBounds, wanderSpeed, wanderSlope, wanderRoundness,
wanderTurnToForwardRatio, wanderCycleRate, wanderCycleStandardDeviation,
wanderSpeedStandardDeviation, wanderPhase); // Start wandering
}

// Placeholder loop function, required for Arduino structure but not used
void _loop()
{
}

// Main loop function, continuously called by Arduino framework
void loop()
{
    _loop(); // Call the placeholder loop function
}

```

A.2. Blink Base Behavior

```

#include <Arduino.h> // Core Arduino library for basic functions and macros
#include <MeMCore.h> // Library for Makeblock electronic modules like motors and sensors
#include <SoftwareSerial.h> // Library for serial communication on digital pins
#include <Wire.h> // Library for I2C communication
#include <stdlib.h> // Standard library for general operations like random numbers
#include <math.h> // Math library for advanced mathematical operations

// Timer control variables to track the current and last time measurements
double currentTime = 0;
double lastTime = 0;

// Initialize an RGB LED module connected to port 7 on the main board
MeRGBLed rgbled_7(7, 2);

// Variables for controlling the blinking behavior of the LED

```

```

double lightsOnDuration; // How long the lights stay on during a cycle
double lightsOffDuration; // How long the lights stay off during a cycle
double targetRedIntensity; // Target intensity for the red component
double targetGreenIntensity; // Target intensity for the green component
double targetBlueIntensity; // Target intensity for the blue component
int brightness; // Current brightness level
int blinkCycle; // Counter for the number of blink cycles

// Boolean flag to check if the input parameters for blinking are valid
boolean doBlink;

// General input parameters for controlling the overall behavior duration
double duration = 10; // Duration for the blink behavior

// Input parameters specifically for the blinking behavior
double blinkTemperature = 0.9; // Determines the color temperature for the LED
double blinkSlope = 1; // Determines how the intensity changes over time
double blinkLightsOnToOffRatio = 0.9; // Ratio of on-time to off-time
double blinkCycleRate = 2; // How many cycles per second
double blinkCycleStandardDeviation = 0.5; // Variability in the cycle rate
double blinkTemperatureStandardDeviation = 0.1; // Variability in the color temperature
double blinkPhase = 0; // Initial phase delay before starting to blink

// Function to calculate elapsed time since the last reset
double getLastTime()
{
    return currentTime = millis() / 1000.0 - lastTime;
}

// Generates a Gaussian-distributed random number based on standard deviation
double GenerateGaussian(double standardDeviation)
{
    double u1 = rand() / (RAND_MAX + 1.0);
    double u2 = rand() / (RAND_MAX + 1.0);
    double z0 = sqrt(-2.0 * log(u1)) * cos(2.0 * M_PI * u2);
    return z0 * standardDeviation;
}

// Generates a random number within a specified range
double GetRandomNumber(double min, double max)
{
    double randomNumber = (double)rand() / RAND_MAX;
    return (randomNumber * (max - min)) + min;
}

// Caps a number to be within a specified range
void CapNumber(double* number, double lowerLimit, double upperLimit)
{
    if (*number < lowerLimit) {
        *number = lowerLimit;
    }
    if (*number > upperLimit) {
        *number = upperLimit;
    }
}

// Checks if the input parameters for the blinking behavior are within valid ranges
void CheckValidBlinkInput(double temperature, double slope, double lightsOnToOffRatio,
    double cycleRate, double cycleStandardDeviation, double temperatureStandardDeviation,
    double phase)
{
    boolean isValidTemperature = temperature >= 0 && temperature <= 1;
    boolean isValidSlope = slope >= -5 && slope <= 5;
    boolean isValidLightsOnToOffRatio = lightsOnToOffRatio <= 1 && lightsOnToOffRatio > 0;
    boolean isCycleRatePositive = cycleRate > 0;
    boolean isValidStandardDeviation = cycleStandardDeviation >= 0 &&
        temperatureStandardDeviation >= 0;
    boolean isValidPhase = phase >= 0;

    // If all conditions are met, the input is considered valid
    doBlink = isValidTemperature && isValidSlope && isValidLightsOnToOffRatio &&
        isCycleRatePositive && isValidStandardDeviation && isValidPhase;
}

```

```

// Sets the durations for the lights being on and off based on the input parameters
void SetBlinkDurations(double lightsOnToOffRatio, double cycleRate, double
    cycleStandardDeviation)
{
    lightsOffDuration = (1 - lightsOnToOffRatio) / cycleRate;
    lightsOffDuration += GenerateGaussian(cycleStandardDeviation);
    CapNumber(&lightsOffDuration, 0, 1 / cycleRate);
    lightsOnDuration = 1 / cycleRate - lightsOffDuration;
}

// Determines the target intensities for the RGB components of the LED based on
    temperature
void SetTargetIntensities(double temperature, double temperatureStandardDeviation)
{
    double targetTemperature = temperature + GenerateGaussian(temperatureStandardDeviation);
    CapNumber(&targetTemperature, 0, 1);

    // Adjust the RGB intensities based on the calculated target temperature
    if (targetTemperature == 0.5) {
        targetRedIntensity = 200;
        targetGreenIntensity = 255;
        targetBlueIntensity = 0;
    } else if (targetTemperature > 0.5) {
        targetRedIntensity = round(110 * targetTemperature + 145);
        targetGreenIntensity = round(-510 * targetTemperature + 510);
        targetBlueIntensity = 0;
    } else if (targetTemperature < 0.5) {
        targetRedIntensity = round(100 * targetTemperature);
        targetGreenIntensity = round(510 * targetTemperature);
        targetBlueIntensity = round(-510 * targetTemperature + 255);
    }
}

// Method to execute the blink base behavior
void Blink(double duration,
    double blinkTemperature, double blinkSlope, double blinkLightsOnToOffRatio,
    double blinkCycleRate, double blinkCycleStandardDeviation, double
    blinkTemperatureStandardDeviation, double blinkPhase)
{
    // INPUT CHECKS AND VARIABLE INITIALIZATION

    // Validates the input parameters to ensure they are within acceptable ranges for the
        blink behavior.
    CheckValidBlinkInput(blinkTemperature, blinkSlope, blinkLightsOnToOffRatio,
        blinkCycleRate, blinkCycleStandardDeviation, blinkTemperatureStandardDeviation,
        blinkPhase);

    // Proceeds only if the input parameters are validated successfully.
    if (doBlink) {
        // Sets the color intensities for the RGB LED based on the calculated temperature,
            including randomness.
        SetTargetIntensities(blinkTemperature, blinkTemperatureStandardDeviation);

        // Determines the durations for which the LED will stay on and off during each blink
            cycle, including randomness.
        SetBlinkDurations(blinkLightsOnToOffRatio, blinkCycleRate, blinkCycleStandardDeviation
            );

        // Initializes the counter that keeps track of the number of completed blink cycles.
        blinkCycle = 0;

        // Initializes the brightness adjustment factor; used when blinkSlope != 0 to vary
            intensity.
        brightness = 1;
    }

    // MAIN LOOP

    // Resets the timer to keep track of the blink behavior's duration.
    lastTime = millis() / 1000.0;

    // Continues blinking for the specified duration.

```

```

while (!(getLastTime() > duration)) {
    // Placeholder for tasks that need continuous execution within the loop.

    // Checks again if blink behavior should continue based on the doBlink flag.
    if (doBlink) {
        // Delays the start of blinking until after the specified phase delay.
        if (getLastTime() - blinkPhase >= 0) {
            // Handles constant intensity blinking without any slope for intensity change.
            if (blinkSlope == 0) {
                // Checks if it's time to turn the lights on within the current blink cycle.
                if (getLastTime() - blinkPhase < blinkCycle / blinkCycleRate + lightsOnDuration)
                {
                    // Sets the LED color using the target intensity values.
                    rgbled_7.setColor(0, targetRedIntensity, targetGreenIntensity,
targetBlueIntensity);
                    rgbled_7.show();
                }
                // Handles increasing intensity blinking when blinkSlope is positive.
                else if (blinkSlope > 0) {
                    // Gradually increases brightness until it reaches the maximum value.
                    if (getLastTime() - blinkPhase < blinkCycle / blinkCycleRate + brightness *
lightsOnDuration / 100) {
                        // Adjusts the LED color intensity based on the current brightness.
                        rgbled_7.setColor(0, round(blinkSlope * brightness * targetRedIntensity / 100)
,
                                round(blinkSlope * brightness * targetGreenIntensity /
100),
                                round(blinkSlope * brightness * targetBlueIntensity /
100));
                        rgbled_7.show();
                    } else {
                        // Increases the brightness for the next iteration, if it has not reached the
maximum.
                        if (brightness < 100) {
                            brightness += 1;
                        }
                    }
                }
                // Handles decreasing intensity blinking when blinkSlope is negative.
                else if (blinkSlope < 0) {
                    // Gradually decreases brightness until it reaches the minimum value.
                    if (getLastTime() - blinkPhase < blinkCycle / blinkCycleRate + brightness *
lightsOnDuration / 100) {
                        // Adjusts the LED color intensity based on the current brightness.
                        rgbled_7.setColor(0, round(blinkSlope * (brightness - 100) *
targetRedIntensity / 100),
                                round(blinkSlope * (brightness - 100) *
targetGreenIntensity / 100),
                                round(blinkSlope * (brightness - 100) *
targetBlueIntensity / 100));
                        rgbled_7.show();
                    } else {
                        // Increases the brightness for the next iteration, if it has not reached the
maximum.
                        if (brightness < 100) {
                            brightness += 1;
                        }
                    }
                }
            }

            // Turns the lights off after the on-duration within the current cycle.
            if (getLastTime() - blinkPhase > blinkCycle / blinkCycleRate + lightsOnDuration &&
getLastTime() - blinkPhase < (blinkCycle + 1) / blinkCycleRate) {
                rgbled_7.setColor(0, 0, 0, 0);
                rgbled_7.show();
            }

            // Prepares for the next blink cycle once the current cycle completes.
            if (getLastTime() - blinkPhase > (blinkCycle + 1) / blinkCycleRate) {
                // Increments the blink cycle counter to track the number of completed cycles.
                blinkCycle += 1;
            }
        }
    }
}

```

```

        // Resets the brightness to its initial value for the next cycle.
        brightness = 1;

        // Re-calculates the target color intensities for the next cycle to include
        variability.
        SetTargetIntensities(blinkTemperature, blinkTemperatureStandardDeviation);

        // Re-calculates the durations for lights on and off for the next cycle to
        include variability.
        SetBlinkDurations(blinkLightsOnToOffRatio, blinkCycleRate,
        blinkCycleStandardDeviation);
    }
}
}

// Turns off the LED at the end of the blinking behavior to ensure it does not stay on.
rgbled_7.setColor(0, 0, 0, 0);
rgbled_7.show();
}

void setup()
{
    // Initial setup for the RGB LED.
    rgbled_7.fillPixelsBak(0, 2, 1); // Pre-configure the LED with a base color or pattern.
    randomSeed(0); // Initialize the random number generator seed.

    // Start the blinking behavior with the specified parameters.
    Blink(duration, blinkTemperature, blinkSlope, blinkLightsOnToOffRatio, blinkCycleRate,
    blinkCycleStandardDeviation, blinkTemperatureStandardDeviation, blinkPhase);
}

// Placeholder loop function, required for Arduino structure but not used
void _loop()
{
}

void loop()
{
    _loop(); // Invoke the placeholder loop function within the main loop.
}

```

A.3. Beep Base Behavior

```

#include <Arduino.h> // Core Arduino library for basic input/output functions, types, and
constants.
#include <MeMCore.h> // Library for Makeblock electronic modules, including the buzzer.
#include <SoftwareSerial.h> // Library for serial communication on digital pins.
#include <Wire.h> // Library for I2C communication.
#include <stdlib.h> // Standard library for general utility functions.
#include <math.h> // Library for mathematical functions.

// Timer control variables to measure elapsed time.
double currentTime = 0; // Current time in seconds since the start.
double lastTime = 0; // Time at the last significant event to calculate elapsed time.

// Arduino component for sound generation.
MeBuzzer buzzer; // Buzzer object for emitting sounds.

// Beep control variables for managing beep pattern.
double soundDuration; // Duration of the sound in each beep cycle.
double silenceDuration; // Duration of silence in each beep cycle.
double targetPitch; // Desired pitch of the beep in Hertz.
double currentPitch; // Current pitch of the beep in Hertz, used during execution.
double semitone; // Incremental change in pitch between beeps, related to musical
semitones.
int beepCycle; // Counter for the number of completed beep cycles.

// Flag to determine if the beep input parameters are valid.

```

```

boolean doBeep;

// General input parameter for the behavior's duration.
double duration = 10; // Duration for the beep behavior in seconds.

// Specific input parameters for beep behavior customization.
double beepPitch = 400; // Base pitch of the beep in Hertz.
double beepSlope = 1; // Determines how the pitch changes over time.
double beepSoundToSilenceRatio = 0.9; // Ratio of sound duration to silence duration in
    each cycle.
double beepCycleRate = 2; // Number of beep cycles per second.
double beepCycleStandardDeviation = 0.5; // Variability in the cycle rate to introduce
    randomness.
double beepPitchStandardDeviation = 100; // Variability in the beep pitch to introduce
    randomness.
double beepRandomSoundProbability = 0.3; // Probability of playing a random sound instead
    of the target pitch.
double beepPhase = 0; // Initial phase delay before starting the beep behavior.

// Function to calculate elapsed time since the program started in seconds.
double getLastTime()
{
    return currentTime = millis() / 1000.0 - lastTime;
}

// Custom delay function to pause execution for a specified number of seconds.
void _delay(double seconds)
{
    long endTime = millis() + seconds * 1000; // Calculate end time in milliseconds.
    while (millis() < endTime) _loop(); // Wait until the end time is reached.
}

// Generates a Gaussian-distributed random number based on standard deviation.
double GenerateGaussian(double standardDeviation)
{
    double u1 = rand() / (RAND_MAX + 1.0); // Generate uniform random number u1.
    double u2 = rand() / (RAND_MAX + 1.0); // Generate uniform random number u2.
    double z0 = sqrt(-2.0 * log(u1)) * cos(2.0 * M_PI * u2); // Box-Muller transform for
        Gaussian distribution.
    return z0 * standardDeviation;
}

// Generates a random number within the specified range [min, max].
double GetRandomNumber(double min, double max)
{
    double randomNumber = (double)rand() / RAND_MAX; // Generate a uniform random number
        between 0 and 1.
    return (randomNumber * (max - min)) + min; // Scale and shift the number to the
        specified range.
}

// Caps a number to be within the specified lower and upper limits.
void CapNumber(double* number, double lowerLimit, double upperLimit)
{
    if (*number < lowerLimit) {
        *number = lowerLimit; // Set to lower limit if below it.
    }
    if (*number > upperLimit) {
        *number = upperLimit; // Set to upper limit if above it.
    }
}

// Validates the input parameters for the beep behavior.
void CheckValidBeepInput(double pitch, double slope, double soundToSilenceRatio, double
    cycleRate, double cycleStandardDeviation, double pitchStandardDeviation, double
    randomSoundProbability, double phase)
{
    // Check each parameter against its valid range.
    boolean isValidPitch = pitch >= 80 && pitch <= 3000; // Valid pitch range.
    boolean isValidSlope = (slope < 0 && slope >= log(40 / pitch) / log(2)) || (slope > 0 &&
        slope <= log(6000 / pitch) / log(2)) || slope == 0; // Valid slope conditions.
    boolean isValidSoundToSilenceRatio = soundToSilenceRatio <= 1 && soundToSilenceRatio >=
        0; // Valid ratio range.
}

```

```

boolean isCycleRatePositive = cycleRate > 0; // Cycle rate must be positive.
boolean isValidStandardDeviation = cycleStandardDeviation >= 0 && pitchStandardDeviation
    >= 0; // Standard deviations must be non-negative.
boolean isValidRandomSoundProbability = randomSoundProbability <= 1 &&
    randomSoundProbability >= 0; // Valid probability range.
boolean isValidPhase = phase >= 0; // Phase must be non-negative.

// If all conditions are met, the input is considered valid.
doBeep = isValidPitch && isValidSlope && isValidSoundToSilenceRatio &&
    isCycleRatePositive && isValidStandardDeviation && isValidRandomSoundProbability &&
    isValidPhase;
}

// Sets the durations for sound and silence based on the ratio and cycle rate.
void SetBeepDurations(double soundToSilenceRatio, double cycleRate, double
    cycleStandardDeviation)
{
    silenceDuration = (1 - soundToSilenceRatio) / cycleRate; // Calculate base silence
        duration.
    silenceDuration += GenerateGaussian(cycleStandardDeviation); // Add Gaussian randomness.
    CapNumber(&silenceDuration, 0, 1 / cycleRate); // Ensure the duration is within valid
        bounds.

    soundDuration = 1 / cycleRate - silenceDuration; // Calculate sound duration based on
        the remaining time.
}

// Sets the target pitch for the beep, including randomness.
void SetTargetPitch(double pitch, double pitchStandardDeviation)
{
    targetPitch = pitch + GenerateGaussian(pitchStandardDeviation); // Add randomness to the
        pitch.
    CapNumber(&targetPitch, 80, 3000); // Ensure pitch is within a valid range.

    currentPitch = targetPitch; // Initialize current pitch to target pitch for the start of
        behavior.
}

// Plays a sound or silence based on a given probability.
void PlayRandomSoundWithProbability(double slope, double randomSoundProbability, double
    pitchStandardDeviation)
{
    double randomNumber = GetRandomNumber(0, 1); // Generate a random number to compare
        against probability.

    double randomPitch; // Variable for the pitch of the random sound.
    if (slope == 0) {
        randomPitch = targetPitch + GenerateGaussian(pitchStandardDeviation); // Generate
            random pitch variation.
    } else {
        int randomSemitone = (int)GetRandomNumber(1, 12); // Random semitone for pitch
            variation.
        randomPitch = exp(log(targetPitch) + slope * randomSemitone / 12 * log(2)); //
            Calculate pitch based on semitone change.
    }

    if (randomNumber < randomSoundProbability) {
        buzzer.tone(randomPitch, silenceDuration * 1000); // Play the random sound if within
            probability.
    } else {
        _delay(silenceDuration); // Otherwise, maintain silence for the duration.
    }
}

// Method to execute the beep base behavior
void Beep(double duration,
    double beepPitch, double beepSlope, double beepSoundToSilenceRatio, double
    beepCycleRate, double beepCycleStandardDeviation, double beepPitchStandardDeviation,
    double beepRandomSoundProbability, double beepPhase)
{
    // INPUT CHECKS AND VARIABLE INITIALIZATION

    // Check if beep input is valid

```



```

CheckValidBeepInput(beepPitch, beepSlope, beepSoundToSilenceRatio, beepCycleRate,
    beepCycleStandardDeviation, beepPitchStandardDeviation, beepRandomSoundProbability,
    beepPhase);

if (doBeep) {
    // Set the target pitch based on given input
    SetTargetPitch(beepPitch, beepPitchStandardDeviation);

    // Set sound and silence durations based on given input
    SetBeepDurations(beepSoundToSilenceRatio, beepCycleRate, beepCycleStandardDeviation);

    // Variable used to control the beep cycles
    beepCycle = 0;

    // Variable used to compute the pitch of each note
    semitone = 1;
}

// MAIN LOOP

// Initialize timer
lastTime = millis() / 1000.0;

// Loop during given duration
while (!(getLastTime() > duration)) {
    _loop();

    // If valid beep input is given
    if (doBeep) {
        // Start executing after the given phase
        if (getLastTime() - beepPhase >= 0) {
            // Check if the beep slope is zero (constant pitch)
            if (beepSlope == 0) {
                if (getLastTime() - beepPhase < beepCycle / beepCycleRate + soundDuration) {
                    // Play the sound at the current pitch for the given duration
                    buzzer.tone(currentPitch, soundDuration * 1000);
                }
            }

            // Check if the beep slope is positive (rising pitch)
            if (beepSlope > 0) {
                if (getLastTime() - beepPhase < beepCycle / beepCycleRate + semitone *
                    soundDuration / 12) {
                    // Play the sound at the current pitch for the given duration
                    buzzer.tone(currentPitch, soundDuration / 12 * 1000);
                } else {
                    // Change current pitch to the next rising semitone
                    currentPitch = exp(log(targetPitch) + beepSlope * semitone / 12 * log(2));

                    // Increase semitone by one
                    if (semitone < 12) {
                        semitone += 1;
                    }
                }
            }
        }

        // Check if the beep slope is negative (falling pitch)
        if (beepSlope < 0) {
            if (getLastTime() - beepPhase < beepCycle / beepCycleRate + semitone *
                soundDuration / 12) {
                // Play the sound at the current pitch for the given duration
                buzzer.tone(currentPitch, soundDuration / 12 * 1000);
            } else {
                // Change current pitch to the next falling semitone
                currentPitch = exp(log(targetPitch) + beepSlope * semitone / 12 * log(2));

                // Increase semitone by one
                if (semitone < 12) {
                    semitone += 1;
                }
            }
        }
    }
}

```

```

    // Play either a random note or silence based on the given probability for the
    // silence duration
    if (getLastTime() - beepPhase > beepCycle / beepCycleRate + soundDuration &&
        getLastTime() - beepPhase < (beepCycle + 1) / beepCycleRate) {
        PlayRandomSoundWithProbability(beepSlope, beepRandomSoundProbability,
        beepPitchStandardDeviation);
    }

    // Check if the beep cycle has finished
    if (getLastTime() - beepPhase > (beepCycle + 1) / beepCycleRate) {
        // Increase the count of cycles by one
        beepCycle += 1;

        // Reset semitone to one
        semitone = 1;

        // Change the target pitch
        SetTargetPitch(beepPitch, beepPitchStandardDeviation);

        // Change sound and silence durations
        SetBeepDurations(beepSoundToSilenceRatio, beepCycleRate,
        beepCycleStandardDeviation);
    }
}
}
}

// Setup function to initialize the robot and start the wander behavior
void setup()
{
    randomSeed(0); // Initialize the random number generator seed.

    // Start the beeping behavior with the specified parameters.
    Beep(duration, beepPitch, beepSlope, beepSoundToSilenceRatio, beepCycleRate,
        beepCycleStandardDeviation, beepPitchStandardDeviation, beepRandomSoundProbability,
        beepPhase);
}

// Placeholder loop function, required for Arduino structure but not used
void _loop()
{
}

// Main loop function, continuously called by Arduino framework
void loop()
{
    _loop(); // Call the placeholder loop function
}

```

A.4. Sobol Sequences Sampling

```

import pandas as pd
from scipy.stats import qmc

# Define the number of variables and samples
n_vars = 9
n_samples = 2 ** 9

# Initialize the Sobol sequence generator
sobol_sampler = qmc.Sobol(d=n_vars, scramble=True)

# Generate samples
sobol_samples = sobol_sampler.random(n=n_samples)

# Define the lower bounds of the samples
l_bounds = [
    30, # wanderSpeed

```

```

    0,      # wanderRoundness
    0.5,    # wanderCycleRate
    0,      # blinkTemperature
    -1,     # blinkSlope
    0.5,    # blinkCycleRate
    100,    # beepPitch
    -1,     # beepSlope
    0.5     # beepCycleRate
]

# Define the upper bounds of the samples
u_bounds = [
    100,    # wanderSpeed
    1,      # wanderRoundness
    6,      # wanderCycleRate
    1,      # blinkTemperature
    1,      # blinkSlope
    6,      # blinkCycleRate
    1000,   # beepPitch
    1,      # beepSlope
    6       # beepCycleRate
]

# Scale samples to the given range
sobol_samples_scaled = qmc.scale(sobol_samples, l_bounds, u_bounds)

# Define the column names of the dataframe
column_names = ["wanderSpeed", "wanderRoundness", "wanderCycleRate",
                "blinkTemperature", "blinkSlope", "blinkCycleRate",
                "beepPitch", "beepSlope", "beepCycleRate"]

# Convert the generated scaled sobol samples into a dataframe
df_sobol = pd.DataFrame(data=sobol_samples_scaled, columns=column_names)

# Round values of "beepSlope" and "blinkSlope" columns to the nearest integer
df_sobol.beepSlope = round(df_sobol.beepSlope)
df_sobol.blinkSlope = round(df_sobol.blinkSlope)

# Save samples to CSV
df_sobol.to_csv("samples.csv", sep=',', header=False, index=False)

```

A.5. Analysis

```

#!/usr/bin/env python
# coding: utf-8

# In[ ]:

import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import scipy.stats as stats
import pickle
from itertools import combinations
from sklearn.preprocessing import StandardScaler, PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE
from sklearn.mixture import GaussianMixture
from sklearn.metrics import silhouette_score
from sklearn.feature_selection import f_regression
from sklearn.model_selection import KFold, GridSearchCV
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from statsmodels.stats.multitest import multipletests

```

```

# In[ ]:

custom_params = {"axes.spines.right": False, "axes.spines.top": False}
sns.set_theme(style="ticks", rc=custom_params)

# In[ ]:

# Define the emotion intensities columns
intensity_columns = [
    'joy_intensity', 'sadness_intensity', 'anger_intensity',
    'fear_intensity', 'disgust_intensity', 'surprise_intensity'
]

# Define the SAM columns
sam_columns = ['pleasure', 'arousal', 'dominance']

# Define the dependent variables
dependent_vars = intensity_columns + sam_columns

# Define the index of each dependent variables in the y dataset
dependent_var_index = {dependent_var: i for i, dependent_var in enumerate(dependent_vars)}

# Define the independent variables continuous independent variables
independent_vars = [
    'wander_speed', 'wander_roundness', 'wander_cycle_rate',
    'blink_temperature', 'blink_slope', 'blink_cycle_rate',
    'beep_pitch', 'beep_slope', 'beep_cycle_rate'
]

# Specify which of the independent variables are continuous
independent_continuous_vars = [
    'wander_speed', 'wander_roundness', 'wander_cycle_rate',
    'blink_temperature', 'blink_cycle_rate',
    'beep_pitch', 'beep_cycle_rate'
]

# In[ ]:

titles = {'wander_speed': 'Wander Speed', 'wander_roundness': 'Wander Roundness', '
wander_cycle_rate': 'Wander Cycle Rate',
'blink_temperature': 'Blink Temperature', 'blink_slope': 'Blink Slope', '
blink_cycle_rate': 'Blink Cycle Rate',
'beep_pitch': 'Beep Pitch', 'beep_slope': 'Beep Slope', 'beep_cycle_rate': 'Beep
Cycle Rate',
'joy_intensity': 'Joy Intensity', 'sadness_intensity': 'Sadness Intensity', '
anger_intensity': 'Anger Intensity',
'fear_intensity': 'Fear Intensity', 'disgust_intensity': 'Disgust Intensity', '
surprise_intensity': 'Surprise Intensity',
'pleasure': 'Pleasure', 'arousal': 'Arousal', 'dominance': 'Dominance',
'cluster_prob_0': 'Membership Probability of Cluster 1', 'cluster_prob_1': '
Membership Probability of Cluster 2',
'cluster_prob_2': 'Membership Probability of Cluster 3', 'cluster_prob_3': '
Membership Probability of Cluster 4',
'cluster_prob_4': 'Membership Probability of Cluster 5', 'cluster_prob_5': '
Membership Probability of Cluster 6',
'cluster_prob_6': 'Membership Probability of Cluster 7', 'cluster_prob_7': '
Membership Probability of Cluster 8',
'cluster_prob_8': 'Membership Probability of Cluster 9',
'lin_model': 'Linear trained with X', 'lin_int_model': 'Linear trained with
X_int',
'lin_gmm_model': 'Linear trained with X_gmm', 'lin_gmm_int_model': 'Linear
trained with X_gmm_int',
'rf_model': 'Random forest trained with X', 'rf_int_model': 'Random forest
trained with X_int',
'rf_gmm_model': 'Random forest trained with X_gmm', 'rf_gmm_int_model': 'Random
forest trained with X_gmm_int',
'mse_scores': 'MSE', 'r2_scores': '$R^2$'}

```

```

# In[ ]:

# Create the interaction term names in the titles dictionary
keys = np.array(list(titles.keys()))
mask = ['wander' in key or 'blink' in key or 'beep' in key or 'cluster' in key for key in
        titles.keys()]

# Get all combinations of 2 variables
variable_combinations = list(combinations(keys[mask], 2))

# Create new titles for interaction terms
for var1, var2 in variable_combinations:
    if f"{var1} {var2}" not in titles and f"{var2} {var1}" not in titles:
        new_title = '{} × {}'.format(titles[var1], titles[var2])
        titles['{} {}'.format(var1, var2)] = new_title

# In[ ]:

score_types = ['mse_scores', 'r2_scores']
model_types = ['lin_model', 'lin_int_model', 'lin_gmm_model', 'lin_gmm_int_model', '
               rf_model', 'rf_int_model', 'rf_gmm_model', 'rf_gmm_int_model']

# ## Modelling

# ### Hypotheses Testing

# In[ ]:

def spearman_test(df, independent_vars, dependent_vars, alpha=0.05):
    data = []
    p_values_non_zero = []
    p_values_positive = []
    p_values_negative = []

    for independent_var in independent_vars:
        for dependent_var in dependent_vars:
            # Calculate Spearman correlation for two-sided test
            rho, p_non_zero = stats.spearmanr(df[independent_var], df[dependent_var],
            alternative='two-sided')

            # Calculate Spearman correlation for one-sided tests
            _, p_negative = stats.spearmanr(df[independent_var], df[dependent_var],
            alternative='less')
            _, p_positive = stats.spearmanr(df[independent_var], df[dependent_var],
            alternative='greater')

            # Collect p-values for later adjustment
            p_values_non_zero.append(p_non_zero)
            p_values_positive.append(p_positive)
            p_values_negative.append(p_negative)

            # Store the initial results
            row = {'independent_variable': independent_var, 'dependent_variable':
            dependent_var, 'correlation': rho,
                   'p_non_zero': p_non_zero, 'is_non_zero': p_non_zero < alpha,
                   'p_positive': p_positive, 'is_positive': p_positive < alpha / 2,
                   'p_negative': p_negative, 'is_negative': p_negative < alpha / 2}

            data.append(row)

    # Convert the list of dictionaries to a DataFrame
    spearman_test_results_df = pd.DataFrame(data)

    spearman_test_results_df.set_index(['independent_variable', 'dependent_variable'],
    inplace=True)

    return spearman_test_results_df

```

```

# In[ ]:

def get_correlation_annots(spearman_test_results_df):
    correlations = spearman_test_results_df.correlation.to_numpy()
    annot = correlations.astype(str)
    indexed_spearman_test_results_df = spearman_test_results_df.reset_index()
    significance_mask = indexed_spearman_test_results_df[indexed_spearman_test_results_df[
        'is_non_zero'] & (indexed_spearman_test_results_df['is_positive'] |
        indexed_spearman_test_results_df['is_negative'])].index

    # Apply asterisks to significant correlations
    for i in range(len(correlations)):
        if i in significance_mask:
            annot[i] = f'{correlations[i]:.3f}*'
        else:
            annot[i] = f'{correlations[i]:.3f}'

    annot = annot.reshape(9, 9).T

    return annot

# In[ ]:

def t_test(df, score_types, model_types, dependent_vars, names, alpha=0.05):
    data = []
    p_values_unequal = []
    p_values_positive = []
    p_values_negative = []

    for score_type in score_types:
        for model_type in model_types:
            for dependent_var in dependent_vars:
                # Get the score and baseline score
                score = df.loc[score_type, dependent_var, model_type][names[0]]
                baseline_score = df.loc[score_type, dependent_var, model_type][names[1]]

                # Calculate one-sample t-test for two-sided hypothesis
                t_stat, p_unequal = stats.ttest_rel(score, baseline_score)

                # Calculate one-sample t-test for one-sided hypotheses
                t_stat, p_positive = stats.ttest_rel(score, baseline_score, alternative='
greater')
                t_stat, p_negative = stats.ttest_rel(score, baseline_score, alternative='
less')

                # Collect p-values for later adjustment
                p_values_unequal.append(p_unequal)
                p_values_positive.append(p_positive)
                p_values_negative.append(p_negative)

                # Store the initial results
                row = {'model_type': model_type, 'score_type': score_type, '
dependent_variable': dependent_var,
                    'baseline_scores': baseline_score, 'scores': score, 't_statistic':
t_stat,
                    'p_unequal': p_unequal, 'is_unequal_significant': p_unequal < alpha
,
                    'p_positive': p_positive, 'is_positive_significant': p_positive <
alpha / 2,
                    'p_negative': p_negative, 'is_negative_significant': p_negative <
alpha / 2}

                data.append(row)

    # Convert the list of dictionaries to a DataFrame
    t_test_results_df = pd.DataFrame(data)

```

```

t_test_results_df.set_index(['score_type', 'dependent_variable', 'model_type'],
                             inplace=True)

return t_test_results_df

# In[ ]:

def bonferroni_correction(df, alpha=0.05, apply_correction=True):
    # Apply multiple testing correction
    statistics = df['statistic'].to_numpy()
    p_values = df['p_value'].to_numpy()
    if apply_correction:
        test_results = multipletests(p_values, alpha=alpha, method='bonferroni')
        data = {}
        data['statistic'] = statistics
        data['p_value'] = p_values
        if apply_correction:
            data['p_value_corrected'] = test_results[1]
            data['reject_h0'] = test_results[0]
        else:
            data['reject_h0'] = data['p_value'] < alpha
    test_results_df = pd.DataFrame(data)
    test_results_df.index = df.index
    return test_results_df

# ### Model Training

# In[ ]:

def backward_selection(X, y, significance_level=0.05):
    included = list(X.columns)
    p_values_dict = {}

    # Calculate initial p-values for all features
    all_p_values = f_regression(X, y)[1]
    for I, column in enumerate(X.columns):
        p_values_dict[column] = all_p_values[I]

    while True:
        changed = False

        # Fit the model with the currently included features
        model = LinearRegression().fit(X[included], y)
        p_values = pd.Series(f_regression(X[included], y)[1], index=included)

        # Find the feature with the worst p-value
        worst_pval = p_values.max()

        if worst_pval > significance_level:
            worst_feature = p_values.idxmax()
            included.remove(worst_feature)
            changed = True

        if not changed or not included:
            break

    # Convert the p-values dictionary to a DataFrame
    p_values_df = pd.DataFrame.from_dict(p_values_dict, orient='index', columns=['p_value'])

    return included, p_values_df

# In[ ]:

def forward_selection(X, y, significance_level=0.05):
    included = []
    p_values_dict = {}

```

```

# Calculate p-values for all features
all_p_values = f_regression(X, y)[1]
for i, column in enumerate(X.columns):
    p_values_dict[column] = all_p_values[i]

while True:
    changed = False
    excluded = list(set(X.columns) - set(included))

    if not excluded:
        break

    new_pval = pd.Series(index=excluded, dtype=float)

    for new_column in excluded:
        model = LinearRegression().fit(X[included + [new_column]], y)
        p_values = f_regression(X[included + [new_column]], y)[1]
        new_pval[new_column] = p_values[-1] # p-value of the new column

    best_pval = new_pval.min()

    if best_pval < significance_level:
        best_feature = new_pval.idxmin()
        included.append(best_feature)
        changed = True

    if not changed:
        break

# Convert the p-values dictionary to a DataFrame
p_values_df = pd.DataFrame.from_dict(p_values_dict, orient='index', columns=['p_value'
])

return included, p_values_df

# In[ ]:

def k_fold_training_linear_model(X, y, n_splits=5, significance_level=0.05, random_state
=42):
    kf = KFold(n_splits=n_splits, shuffle=True, random_state=random_state)
    train_mse_scores = []
    test_mse_scores = []
    train_r2_scores = []
    test_r2_scores = []
    average_mse_scores = []
    average_r2_scores = []

    best_model = None
    best_selected_features = None
    best_score = float('inf')
    best_train_index = None
    best_test_index = None

    for train_index, test_index in kf.split(X):
        X_train, X_test = X.iloc[train_index], X.iloc[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        selected_features, p_values = backward_selection(X_train, y_train,
significance_level)
        if len(selected_features) == 0:
            selected_features = X_train.columns

        model = LinearRegression().fit(X_train[selected_features], y_train)

        y_train_pred = model.predict(X_train[selected_features])
        y_test_pred = model.predict(X_test[selected_features])
        y_baseline_pred = np.array([y.mean()]) * y_test.shape[0]

        train_mse = mean_squared_error(y_train, y_train_pred)
        test_mse = mean_squared_error(y_test, y_test_pred)

```



```

train_r2 = r2_score(y_train, y_train_pred)
test_r2 = r2_score(y_test, y_test_pred)
average_mse = mean_squared_error(y_test, y_baseline_pred)
average_r2 = r2_score(y_test, y_baseline_pred)

train_mse_scores.append(train_mse)
test_mse_scores.append(test_mse)
train_r2_scores.append(train_r2)
test_r2_scores.append(test_r2)
average_mse_scores.append(average_mse)
average_r2_scores.append(average_r2)

# Update the best model based on the performance on the train set
if train_mse < best_score:
    best_score = train_mse
    best_model = model
    best_selected_features = selected_features
    best_p_values = p_values
    best_train_index = train_index
    best_test_index = test_index

train_mse_scores = np.array(train_mse_scores)
test_mse_scores = np.array(test_mse_scores)
train_r2_scores = np.array(train_r2_scores)
test_r2_scores = np.array(test_r2_scores)
average_mse_scores = np.array(average_mse_scores)
average_r2_scores = np.array(average_r2_scores)

return best_model, best_selected_features, best_p_values, best_train_index,
best_test_index, train_mse_scores, test_mse_scores, train_r2_scores, test_r2_scores,
average_mse_scores, average_r2_scores

# In[ ]:

def k_fold_training_random_forest_model(X, y, param_grid, n_splits=5, random_state=42):
    kf = KFold(n_splits=n_splits, shuffle=True, random_state=random_state)
    train_mse_scores = []
    test_mse_scores = []
    train_r2_scores = []
    test_r2_scores = []
    average_mse_scores = []
    average_r2_scores = []

    best_model = None
    best_score = float('inf')
    best_train_index = None
    best_test_index = None

    for train_index, test_index in kf.split(X):
        X_train, X_test = X[train_index], X[test_index]
        y_train, y_test = y.iloc[train_index], y.iloc[test_index]

        # Use GridSearchCV for hyperparameter tuning
        model = RandomForestRegressor(random_state=random_state)
        grid_search = GridSearchCV(estimator=model, param_grid=param_grid, cv=5, scoring='
neg_mean_squared_error', n_jobs=-1)
        grid_search.fit(X_train, y_train)

        best_fold_model = grid_search.best_estimator_

        y_train_pred = best_fold_model.predict(X_train)
        y_test_pred = best_fold_model.predict(X_test)
        y_baseline_pred = np.array([y.mean()] * y_test.shape[0])

        train_mse = mean_squared_error(y_train, y_train_pred)
        test_mse = mean_squared_error(y_test, y_test_pred)
        train_r2 = r2_score(y_train, y_train_pred)
        test_r2 = r2_score(y_test, y_test_pred)
        average_mse = mean_squared_error(y_test, y_baseline_pred)
        average_r2 = r2_score(y_test, y_baseline_pred)

```

```

train_mse_scores.append(train_mse)
test_mse_scores.append(test_mse)
train_r2_scores.append(train_r2)
test_r2_scores.append(test_r2)
average_mse_scores.append(average_mse)
average_r2_scores.append(average_r2)

# Update the best model based on the performance on the train set
if train_mse < best_score:
    best_score = train_mse
    best_model = best_fold_model
    best_train_index = train_index
    best_test_index = test_index

train_mse_scores = np.array(train_mse_scores)
test_mse_scores = np.array(test_mse_scores)
train_r2_scores = np.array(train_r2_scores)
test_r2_scores = np.array(test_r2_scores)
average_mse_scores = np.array(average_mse_scores)
average_r2_scores = np.array(average_r2_scores)

return best_model, best_train_index, best_test_index, train_mse_scores,
test_mse_scores, train_r2_scores, test_r2_scores, average_mse_scores,
average_r2_scores

# In[ ]:

def print_scores(train_mse_scores, test_mse_scores, train_r2_scores, test_r2_scores):
    print(f"\nAverage Training Mean Squared Error: {train_mse_scores.mean().round(3)}")
    print(f"Average Testing Mean Squared Error: {test_mse_scores.mean().round(3)}")

    print(f"\nAverage Training R2 Score: {train_r2_scores.mean().round(3)}")
    print(f"Average Testing R2 Score: {test_r2_scores.mean().round(3)}")

    best_model_index = np.argmin(train_mse_scores)

    print(f"\n\nTraining Mean Squared Error of the Best Model: {train_mse_scores[
best_model_index].round(3)}")
    print(f"Testing Mean Squared Error of the Best Model: {test_mse_scores[
best_model_index].round(3)}")

    print(f"\nTraining R2 Score of the Best Model: {train_r2_scores[best_model_index].
round(3)}")
    print(f"Testing R2 Score of the Best Model: {test_r2_scores[best_model_index].round(3)
}")

# In[ ]:

def plot_performance(dependent_var, model, model_name, X_train, y_train, X_test, y_test):
    # Define the range of the dependent variable
    if '_intensity' in dependent_var:
        var_range = [0, 5]
    else:
        var_range = [1, 9]

    # Predict on training and test data
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)

    # Plot actual vs predicted values
    plt.figure(figsize=(12, 4))

    # Plot train data
    plt.subplot(1, 2, 1)
    plt.scatter(y_train, y_train_pred, color='royalblue', alpha=0.5, label='Train data')
    plt.plot(var_range, var_range, 'k--', lw=2)
    plt.xlabel(f'Actual {titles[dependent_var]}')
    plt.xlim(var_range[0] - 0.5, var_range[1] + 0.5)
    plt.ylabel(f'Predicted {titles[dependent_var]}')

```

```

plt.ylim(var_range[0] - 0.5, var_range[1] + 0.5)
plt.text(var_range[0], var_range[1] - 0.5, "MSE = {:.3f}".format(mean_squared_error(
y_train, y_train_pred)))
plt.text(var_range[0], var_range[1] - 1, "$R^2$ = {:.3f}".format(r2_score(y_train,
y_train_pred)))
plt.legend(loc="lower right")

# Plot test data
plt.subplot(1, 2, 2)
plt.scatter(y_test, y_test_pred, color='lightskyblue', alpha=0.5, label='Test data')
plt.plot(var_range, var_range, 'k--', lw=2)
plt.xlabel(f'Actual {titles[dependent_var]}')
plt.xlim(var_range[0] - 0.5, var_range[1] + 0.5)
plt.ylabel(f'Predicted {titles[dependent_var]}')
plt.ylim(var_range[0] - 0.5, var_range[1] + 0.5)
plt.text(var_range[0], var_range[1] - 0.5, "MSE = {:.3f}".format(mean_squared_error(
y_test, y_test_pred)))
plt.text(var_range[0], var_range[1] - 1, "$R^2$ = {:.3f}".format(r2_score(y_test,
y_test_pred)))
plt.legend(loc="lower right")

plt.savefig(f'../Data/Figures/{model_name}_performance.pdf', bbox_inches='tight')

plt.show()

# In [ ]:

def plot_residuals(dependent_var, model, model_name, X_train, y_train, X_test, y_test):
    # Define the range of the dependent variable
    if '_intensity' in dependent_var:
        var_range = [0, 5]
    else:
        var_range = [1, 9]

    # Predict on training and test data
    y_train_pred = model.predict(X_train)
    y_test_pred = model.predict(X_test)

    # Plot actual vs predicted values
    plt.figure(figsize=(12, 4))

    # Calculate residuals
    train_residuals = y_train - y_train_pred
    test_residuals = y_test - y_test_pred

    # Plot train residuals
    plt.subplot(1, 2, 1)
    plt.scatter(y_train_pred, train_residuals, color='royalblue', alpha=0.5, label='Train
data')
    plt.axhline(y=0, color='black', linestyle='--')
    plt.xlim(var_range[0] - 0.5, var_range[1] + 0.5)
    plt.xticks(range(var_range[0], var_range[1] + 1))
    plt.ylim(-var_range[1] - 1, var_range[1] + 1)
    plt.yticks(range(-var_range[1], var_range[1] + 1))
    plt.xlabel(f'Predicted {titles[dependent_var]}')
    plt.ylabel('Residuals')
    plt.legend(loc="lower right")

    # Plot test residuals
    plt.subplot(1, 2, 2)
    plt.scatter(y_test_pred, test_residuals, color='lightskyblue', alpha=0.5, label='Test
data')
    plt.axhline(y=0, color='black', linestyle='--')
    plt.xlim(var_range[0] - 0.5, var_range[1] + 0.5)
    plt.xticks(range(var_range[0], var_range[1] + 1))
    plt.ylim(-var_range[1] - 0.5, var_range[1] + 0.5)
    plt.yticks(range(-var_range[1], var_range[1] + 1))
    plt.xlabel(f'Predicted {titles[dependent_var]}')
    plt.ylabel('Residuals')
    plt.legend(loc="lower right")

```

```
plt.savefig(f'../Data/Figures/{model_name}_residuals.pdf', bbox_inches='tight')

plt.tight_layout()
plt.show()

# In[ ]:

def plot_importances(rf_importances_df, plot_name):
    rf_importances_df.index = rf_importances_df.index.map(lambda feature: titles.get(
        feature, feature))
    plt.figure(figsize=(6, 4))
    sns.barplot(x='Importance', y='Feature', data=rf_importances_df.sort_values(by='
    Importance', ascending=False).head(10), color='lightskyblue')
    plt.xlabel('Importance')
    plt.ylabel('')
    plt.savefig(f'../Data/Figures/{plot_name}.pdf', bbox_inches='tight')
    plt.show()

# In[ ]:

def save_model(model, model_name):
    # Save the model to a file using pickle
    with open(f'../Data/Models/{model_name}.pkl', 'wb') as file:
        pickle.dump(model, file)

    print(f"Model {model_name} saved successfully.")

# In[ ]:

def load_model(model_name):
    # Load the model from the file using pickle
    with open(f'../Data/Models/{model_name}.pkl', 'rb') as file:
        loaded_model = pickle.load(file)

    return loaded_model

# ## Data Preparation

# In[ ]:

file_path = '../Data/Processed/rating_numeric.csv'

# In[ ]:

# Load data
data = pd.read_csv(file_path, header=0, index_col=[0, 1])

# Convert start_time and end_time to datetime format
data['start_time'] = pd.to_datetime(data['start_time'])
data['end_time'] = pd.to_datetime(data['end_time'])

# Drop the 'appraisal' column as it is non-numeric
data = data.drop(columns=['appraisal'])

# In[ ]:

# Display basic information about the dataset
data.info()

# In[ ]:
```

```

data.describe(include=np.number)

# In[ ]:

data

# ## Exploratory Data Analysis

# In[ ]:

fig, axes = plt.subplots(3, 3, figsize=(12, 6))

for ax, independent_var in zip(axes.flatten(), independent_vars):
    # Create the histogram in the specified subplot
    sns.histplot(data=data, x=independent_var, ax=ax, color='lightskyblue')
    ax.set_title(titles[independent_var])
    ax.set_xlabel('')
    ax.set_ylabel('Count')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.savefig('../Data/Figures/independent_vars_hists.pdf', bbox_inches='tight')
plt.show()

# In[ ]:

fig, axes = plt.subplots(2, 3, figsize=(12, 6))

for ax, dependent_var in zip(axes.flatten(), intensity_columns):
    # Create the catplot in the specified subplot
    sns.countplot(data=data, x=dependent_var, ax=ax, color='lightskyblue')
    ax.set_title(titles[dependent_var])
    ax.set_xlabel('')
    ax.set_xlim(-0.5, 5.5)
    ax.set_xticks(range(6))
    ax.set_xticklabels(['N/A', 'Very Low', 'Low', 'Average', 'High', 'Very High'])
    ax.set_ylabel('Count')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.savefig('../Data/Figures/intensity_vars_hists.pdf', bbox_inches='tight')
plt.show()

# Next, we create a grid of count plots to visualize the distributions of the Self-
# Assessment Manikin columns:

# In[ ]:

fig, axes = plt.subplots(1, 3, figsize=(12, 3))

for ax, dependent_var in zip(axes.flatten(), sam_columns):
    # Create the catplot in the specified subplot
    sns.countplot(data=data, x=dependent_var, ax=ax, color='lightskyblue')
    ax.set_title(titles[dependent_var])
    ax.set_xlabel('')
    ax.set_xticks(range(9))
    ax.set_xticklabels(range(1,10))
    ax.set_ylabel('Count')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.savefig('../Data/Figures/sam_vars_hists.pdf', bbox_inches='tight')
plt.show()

# #### Correlation Matrix of the Independent Variables

```

```

# In[ ]:

corr_matrix = data[independent_vars + dependent_vars].corr(method='spearman')

# In[ ]:

spearman_test_results_df = spearman_test(data, independent_vars, independent_vars)
annot = get_correlation_annots(spearman_test_results_df)

plt.figure(figsize=(12, 6))
ticklabels = [titles[var] for var in independent_vars]
sns.heatmap(corr_matrix.iloc[:9, :9], mask=np.triu(np.ones_like(corr_matrix.iloc[:9, :9],
    dtype=bool), k=1), annot=annot, xticklabels=ticklabels, yticklabels=ticklabels, cmap='
    coolwarm', fmt='', center=0, vmin=-1, vmax=1, cbar=False)
plt.savefig('../Data/Figures/correlation_independent_vars.pdf', bbox_inches='tight')
plt.show()

# #### Correlation Matrix of the Dependent Variables

# In[ ]:

spearman_test_results_df = spearman_test(data, dependent_vars, dependent_vars)
annot = get_correlation_annots(spearman_test_results_df)

plt.figure(figsize=(12, 6))
ticklabels = [titles[var] for var in dependent_vars]
sns.heatmap(corr_matrix.iloc[9:, 9:], mask=np.triu(np.ones_like(corr_matrix.iloc[9:, 9:],
    dtype=bool), k=1), annot=annot, xticklabels=ticklabels, yticklabels=ticklabels, cmap='
    coolwarm', fmt='', center=0, vmin=-1, vmax=1, cbar=False)
plt.savefig('../Data/Figures/correlation_dependent_vars.pdf', bbox_inches='tight')
plt.show()

# #### Correlation Matrix of the Independent and the Dependent Variables

# In[ ]:

spearman_test_results_df = spearman_test(data, independent_vars, dependent_vars)
annot = get_correlation_annots(spearman_test_results_df)

plt.figure(figsize=(12, 6))
xticklabels = [titles[var] for var in independent_vars]
yticklabels = [titles[var] for var in dependent_vars]
sns.heatmap(corr_matrix.iloc[9:, :9], annot=annot, xticklabels=xticklabels, yticklabels=
    yticklabels, cmap='coolwarm', fmt='', center=0, vmin=-1, vmax=1, cbar=False)
plt.savefig('../Data/Figures/correlation_independent_dependent_vars.pdf', bbox_inches='
    tight')
plt.show()

# ## Outlier Removal

# In[ ]:

# Function to detect and remove outliers using IQR
def remove_outliers(df, columns):
    for col in columns:
        Q1 = df[col].quantile(0.25)
        Q3 = df[col].quantile(0.75)
        IQR = Q3 - Q1
        lower_bound = Q1 - 1.5 * IQR
        upper_bound = Q3 + 1.5 * IQR
        df = df[(df[col] >= lower_bound) & (df[col] <= upper_bound)]
    return df

```

```

# Remove outliers from the dependent variables
cleaned_data = remove_outliers(data, dependent_vars)

# In[ ]:

cleaned_data

# In[ ]:

cleaned_data.groupby("video_id").transform("size").mean()

# ## Exploratory Data Analysis Post Outlier Removal

# In[ ]:

ffig, axes = plt.subplots(3, 3, figsize=(12, 6))

for ax, independent_var in zip(axes.flatten(), independent_vars):
    # Create the histplot in the specified subplot
    sns.histplot(data=cleaned_data, x=independent_var, ax=ax, color='lightskyblue')
    ax.set_title(titles[independent_var])
    ax.set_xlabel('')
    ax.set_ylabel('Count')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.savefig('../Data/Figures/independent_vars_hists_no_outliers.pdf', bbox_inches='tight')
plt.show()

# ### Visualize Distributions of Dependent Variables
#
# Next, we create a grid of count plots to visualize the distributions of the emotion
# intensity variables. This helps us understand the distribution of the intensities for
# each emotion:

# In[ ]:

fig, axes = plt.subplots(2, 3, figsize=(12, 6))

for ax, dependent_var in zip(axes.flatten(), intensity_columns):
    # Create the catplot in the specified subplot
    sns.countplot(data=cleaned_data, x=dependent_var, ax=ax, color='lightskyblue')
    ax.set_title(titles[dependent_var])
    ax.set_xlabel('')
    ax.set_xlim(-0.5, 5.5)
    ax.set_xticks(range(6))
    ax.set_xticklabels(['N/A', 'Very Low', 'Low', 'Average', 'High', 'Very High'])
    ax.set_ylabel('Count')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.savefig('../Data/Figures/intensity_vars_hists_no_outliers.pdf', bbox_inches='tight')
plt.show()

# In[ ]:

fig, axes = plt.subplots(1, 3, figsize=(12, 3))

for ax, dependent_var in zip(axes.flatten(), sam_columns):
    # Create the catplot in the specified subplot
    sns.countplot(data=cleaned_data, x=dependent_var, ax=ax, color='lightskyblue')
    ax.set_title(titles[dependent_var])
    ax.set_xlabel('')
    ax.set_xticks(range(9))
    ax.set_xticklabels(range(1,10))

```

```

    ax.set_ylabel('Count')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.savefig('../Data/Figures/sam_vars_hists_no_outliers.pdf', bbox_inches='tight')
plt.show()

# #### Correlation Matrix of the Independent Variables Post Outlier Removal

# In[ ]:

corr_matrix = cleaned_data[independent_vars + dependent_vars].corr(method='spearman')

# In[ ]:

spearman_test_results_df = spearman_test(cleaned_data, independent_vars, independent_vars)
annot = get_correlation_annots(spearman_test_results_df)

plt.figure(figsize=(12, 6))
ticklabels = [titles[var] for var in independent_vars]
sns.heatmap(corr_matrix.iloc[:9, :9], mask=np.triu(np.ones_like(corr_matrix.iloc[:9, :9],
    dtype=bool), k=1), annot=annot, xticklabels=ticklabels, yticklabels=ticklabels, cmap='
    coolwarm', fmt='', center=0, vmin=-1, vmax=1, cbar=False)
plt.savefig('../Data/Figures/correlation_independent_vars_no_outliers.pdf', bbox_inches='
    tight')
plt.show()

# #### Correlation Matrix of the Dependent Variables Post Outlier Removal

# In[ ]:

spearman_test_results_df = spearman_test(cleaned_data, dependent_vars, dependent_vars)
annot = get_correlation_annots(spearman_test_results_df)

plt.figure(figsize=(12, 6))
ticklabels = [titles[var] for var in dependent_vars]
sns.heatmap(corr_matrix.iloc[9:, 9:], mask=np.triu(np.ones_like(corr_matrix.iloc[9:, 9:],
    dtype=bool), k=1), annot=annot, xticklabels=ticklabels, yticklabels=ticklabels, cmap='
    coolwarm', fmt='', center=0, vmin=-1, vmax=1, cbar=False)
plt.savefig('../Data/Figures/correlation_dependent_vars_no_outliers.pdf', bbox_inches='
    tight')
plt.show()

# #### Correlation Matrix of the Independent and the Dependent Variables Post Outlier
    Removal

# In[ ]:

spearman_test_results_df = spearman_test(cleaned_data, independent_vars, dependent_vars)
annot = get_correlation_annots(spearman_test_results_df)

plt.figure(figsize=(12, 6))
xticklabels = [titles[var] for var in independent_vars]
yticklabels = [titles[var] for var in dependent_vars]
sns.heatmap(corr_matrix.iloc[9:, :9], annot=annot, xticklabels=xticklabels, yticklabels=
    yticklabels, cmap='coolwarm', fmt='', center=0, vmin=-1, vmax=1, cbar=False)
plt.savefig('../Data/Figures/correlation_independent_dependent_vars_no_outliers.pdf',
    bbox_inches='tight')
plt.show()

# ## Reducing Noise in the Dependent Variables by Aggregation

# In[ ]:

```



```

# Aggregate the dependent variables by video_id (mean, median, and mode)
dependent_aggregated_mean = cleaned_data.groupby('video_id')[dependent_vars].mean().
    add_suffix('_mean')
dependent_aggregated_median = cleaned_data.groupby('video_id')[dependent_vars].median().
    add_suffix('_median')
dependent_aggregated_mode = cleaned_data.groupby('video_id')[dependent_vars].agg(lambda x:
    x.mode().iloc[0]).add_suffix('_mode')

# Add the titles of the aggregated variables
for dependent_var in dependent_vars:
    for aggregation in ['mean', 'median', 'mode']:
        dependent_aggregated_var = dependent_var + '_' + aggregation
        titles[dependent_aggregated_var] = aggregation.title() + ' ' + titles[
            dependent_var]

# Merge the aggregated dependent variables with the original independent variables
aggregated_dependent_vars = pd.concat([dependent_aggregated_mean,
    dependent_aggregated_median, dependent_aggregated_mode], axis=1)
cleaned_data = cleaned_data.merge(aggregated_dependent_vars, on='video_id')
cleaned_aggregated_data = cleaned_data.reset_index().drop_duplicates('video_id').set_index
    ('video_id')
cleaned_aggregated_data = cleaned_aggregated_data.drop(['start_time', 'end_time'], axis=1)

# In[ ]:

cleaned_aggregated_data.sort_index()

# In[ ]:

descriptive_statistics = cleaned_aggregated_data[dependent_vars].describe().round(3)
descriptive_statistics.columns = [titles[var] for var in dependent_vars]
descriptive_statistics.loc['mean':, :].T.to_csv('../Data/Tables/dependent_vars_statistics.
    csv')

# In[ ]:

descriptive_statistics

# In[ ]:

descriptive_statistics_aggregated = dependent_aggregated_mean.describe().round(3)
descriptive_statistics_aggregated.columns = [titles[var] for var in dependent_vars]
descriptive_statistics_aggregated.loc['mean':, :].T.to_csv('../Data/Tables/
    dependent_aggregated_vars_statistics.csv')

# In[ ]:

descriptive_statistics_aggregated

# In[ ]:

# Select mean because it's the one that reduces noise the most
target_aggregation = 'mean'

# Select target aggregated variable
mask = [target_aggregation in column for column in cleaned_aggregated_data.columns]
dependent_aggregated_vars = list(cleaned_aggregated_data.columns[mask])

# ## Exploratory Data Analysis Post Aggregation

```

```

# In[ ]:

plt.figure(figsize=(12, 6))
g = sns.pairplot(cleaned_aggregated_data[independent_continuous_vars], diag_kind="hist",
                 plot_kws={'color': 'lightskyblue'}, diag_kws={'color': 'lightskyblue'})
for ax in g.axes.flatten():
    try:
        xlabel = ax.get_xlabel()
        ylabel = ax.get_ylabel()
        if xlabel in titles:
            ax.set_xlabel(titles[xlabel])
        if ylabel in titles:
            ax.set_ylabel(titles[ylabel])
    except:
        continue
plt.savefig('../Data/Figures/independent_vars_pairplot.pdf', bbox_inches='tight')
plt.show()

# In[ ]:

plt.figure(figsize=(10, 6))
g = sns.pairplot(cleaned_aggregated_data[dependent_aggregated_vars], diag_kind="hist",
                 plot_kws={'color': 'lightskyblue'}, diag_kws={'color': 'lightskyblue'})
for ax in g.axes.flatten():
    try:
        xlabel = ax.get_xlabel()
        ylabel = ax.get_ylabel()
        if xlabel in titles:
            ax.set_xlabel(titles[xlabel])
        if ylabel in titles:
            ax.set_ylabel(titles[ylabel])
    except:
        continue
plt.savefig('../Data/Figures/dependent_vars_pairplot.pdf', bbox_inches='tight')
plt.show()

# In[ ]:

fig, axes = plt.subplots(3, 3, figsize=(12, 6))

for ax, independent_var in zip(axes.flatten(), independent_vars):
    # Create the histplot in the specified subplot
    sns.histplot(data=cleaned_aggregated_data, x=independent_var, ax=ax, color='
lightskyblue')
    ax.set_title(titles[independent_var])
    ax.set_xlabel('')
    ax.set_ylabel('Count')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.savefig('../Data/Figures/independent_vars_hists_aggregated.pdf', bbox_inches='tight')
plt.show()

# ### Visualize Distributions of Dependent Variables Post Aggregation

# In[ ]:

fig, axes = plt.subplots(2, 3, figsize=(12, 6))

intensity_columns_aggregated = [column + '_' + target_aggregation for column in
                                intensity_columns]

for ax, dependent_var in zip(axes.flatten(), intensity_columns_aggregated):
    # Create the catplot in the specified subplot
    if 'disgust' in dependent_var:
        sns.histplot(data=cleaned_aggregated_data, x=dependent_var, ax=ax, color='
lightskyblue', bins=5)

```

```

else:
    sns.histplot(data=cleaned_aggregated_data, x=dependent_var, ax=ax, color='
lightskyblue', bins=10)
ax.set_title(titles[dependent_var])
ax.set_xlabel('')
ax.set_xlim(-0.5, 5.5)
ax.set_xticks(range(6))
ax.set_xticklabels(['N/A', 'Very Low', 'Low', 'Average', 'High', 'Very High'])
ax.set_ylabel('Count')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.savefig('../Data/Figures/intensity_vars_hists_aggregated.pdf', bbox_inches='tight')
plt.show()

# In[ ]:

fig, axes = plt.subplots(1, 3, figsize=(12, 3))

sam_columns_aggregated = [column + '_' + target_aggregation for column in sam_columns]

for ax, dependent_var in zip(axes.flatten(), sam_columns_aggregated):
    # Create the catplot in the specified subplot
    sns.histplot(data=cleaned_aggregated_data, x=dependent_var, ax=ax, color='lightskyblue
', bins=10)
ax.set_title(titles[dependent_var])
ax.set_xlabel('')
ax.set_xticks(range(9))
ax.set_xticklabels(range(1,10))
ax.set_ylabel('Count')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.savefig('../Data/Figures/sam_vars_hists_aggregated.pdf', bbox_inches='tight')
plt.show()

# In[ ]:

plt.figure(figsize=(13,3))
sns.boxplot(cleaned_aggregated_data[intensity_columns_aggregated], color='lightskyblue')
plt.xticks(ticks=range(len(intensity_columns_aggregated)),
           labels=[titles[var] for var in intensity_columns_aggregated])
# plt.yticks(ticks=range(6),
#           labels=['N/A', 'Very Low', 'Low', 'Average', 'High', 'Very High'])
plt.yticks(ticks=range(6))
plt.savefig('../Data/Figures/boxplot_intensity_cols.pdf', bbox_inches='tight')
plt.show()

# In[ ]:

plt.figure(figsize=(13,3))
sns.boxplot(cleaned_aggregated_data[sam_columns_aggregated], color='lightskyblue')
plt.xticks(ticks=range(len(sam_columns_aggregated)),
           labels=[titles[var] for var in sam_columns_aggregated])
plt.yticks(ticks=range(1,10))
plt.savefig('../Data/Figures/boxplot_sam_cols.pdf', bbox_inches='tight')
plt.show()

# #### Correlation Matrix of the Independent Variables Post Aggregation

# In[ ]:

corr_matrix = cleaned_aggregated_data[independent_vars + dependent_aggregated_vars].corr(
    method='spearman')

# In[ ]:

```

```

spearman_test_results_df = spearman_test(cleaned_aggregated_data, independent_vars,
                                         independent_vars)
annot = get_correlation_annots(spearman_test_results_df)

plt.figure(figsize=(12, 6))
ticklabels = [titles[var] for var in independent_vars]
sns.heatmap(corr_matrix.iloc[:9, :9], mask=np.triu(np.ones_like(corr_matrix.iloc[:9, :9],
                                                             dtype=bool), k=1), annot=annot, xticklabels=ticklabels, yticklabels=ticklabels, cmap='
coolwarm', fmt='', center=0, vmin=-1, vmax=1, cbar=False)
plt.savefig('../Data/Figures/correlation_independent_vars_aggregated.pdf', bbox_inches='
tight')
plt.show()

# #### Correlation Matrix of the Dependent Variables Post Aggregation

# In[ ]:

spearman_test_results_df = spearman_test(cleaned_aggregated_data,
                                         dependent_aggregated_vars)
annot = get_correlation_annots(spearman_test_results_df)

plt.figure(figsize=(12, 6))
ticklabels = [titles[var] for var in dependent_vars]
sns.heatmap(corr_matrix.iloc[9:, 9:], mask=np.triu(np.ones_like(corr_matrix.iloc[9:, 9:],
                                                             dtype=bool), k=1), annot=annot, xticklabels=ticklabels, yticklabels=ticklabels, cmap='
coolwarm', fmt='', center=0, vmin=-1, vmax=1, cbar=False)
plt.savefig('../Data/Figures/correlation_dependent_vars_aggregated.pdf', bbox_inches='
tight')
plt.show()

# #### Correlation Matrix of the Independent and the Dependent Variables Aggregation

# In[ ]:

spearman_test_results_df = spearman_test(cleaned_data, independent_vars,
                                         dependent_aggregated_vars)
annot = get_correlation_annots(spearman_test_results_df)

plt.figure(figsize=(12, 6))
xticklabels = [titles[var] for var in independent_vars]
yticklabels = [titles[var] for var in dependent_vars]
sns.heatmap(corr_matrix.iloc[9:, :9], annot=annot, xticklabels=xticklabels, yticklabels=
yticklabels, cmap='coolwarm', fmt='', center=0, vmin=-1, vmax=1, cbar=False)
plt.savefig('../Data/Figures/correlation_independent_dependent_vars_aggregated.pdf',
            bbox_inches='tight')
plt.show()

# In[ ]:

# Set up the matplotlib figure
plt.figure(figsize=(16, 4))

# Scatter plot for Wander Speed vs Arousal
plt.subplot(1, 3, 1)
sns.scatterplot(data=cleaned_aggregated_data, x='wander_speed', y='_'.join(('arousal',
target_aggregation)), color='lightskyblue')
plt.xlabel('Wander Speed')
plt.ylabel('Mean Arousal')
plt.ylim(1, 9)

# Scatter plot for Blink Temperature vs Anger Intensity
plt.subplot(1, 3, 2)
sns.scatterplot(data=cleaned_aggregated_data, x='blink_temperature', y='_'.join(('
anger_intensity', target_aggregation)), color='lightskyblue')

```

```

plt.xlabel('Blink Temperature')
plt.ylabel('Mean Anger Intensity')
plt.ylim(0, 5)
plt.yticks(range(0, 6), ['N/A', 'Very Low', 'Low', 'Average', 'High', 'Very High'])

# Scatter plot for Beep Pitch vs Surprise Intensity
plt.subplot(1, 3, 3)
sns.scatterplot(data=cleaned_aggregated_data, x='beep_pitch', y='_'.join(('
    surprise_intensity', target_aggregation)), color='lightskyblue')
plt.xlabel('Beep Pitch')
plt.ylabel('Mean Surprise Intensity')
plt.ylim(0, 5)
plt.yticks(range(0, 6), ['N/A', 'Very Low', 'Low', 'Average', 'High', 'Very High'])

# Display the plots
plt.tight_layout()
plt.show()

### Hypotheses Testing RQ1, RQ2 and RQ3

# In[ ]:

selected = []

# In[ ]:

selected.append(spearman_test_results_df[['correlation', 'p_non_zero']].rename(columns={'
    correlation': 'statistic', 'p_non_zero': 'p_value'}))

# In[ ]:

h1_h2_h3_df = pd.concat(selected, axis=0)

# In[ ]:

h1_h2_h3_df

# In[ ]:

h1_h2_h3_df = bonferroni_correction(h1_h2_h3_df)

# In[ ]:

h1_h2_h3_df

# In[ ]:

h1_h2_h3_df.reset_index(inplace=True)
h1_h2_h3_df['reject_h0'] = h1_h2_h3_df['reject_h0'].astype("string")
h1_h2_h3_df['p_value'] = h1_h2_h3_df['p_value'] / 2
h1_h2_h3_df.drop(columns=['p_value'], inplace=True)
h1_h2_h3_df.loc[h1_h2_h3_df['reject_h0'] == 'True', 'reject_h0'] = 'Reject $H_0$'
h1_h2_h3_df.loc[h1_h2_h3_df['reject_h0'] == 'False', 'reject_h0'] = 'Fail to reject $H_0$'
h1_h2_h3_df.independent_variable = h1_h2_h3_df.independent_variable.map(lambda feature:
    titles.get(feature, feature))
h1_h2_h3_df.dependent_variable = h1_h2_h3_df.dependent_variable.map(lambda feature: titles
    .get(feature, feature))
h1_h2_h3_df.columns = ['Independent Variable', 'Dependent Variable', 'Spearman $\rho$', '
    $p$-value', 'Decision']

```

```

h1_h2_h3_df = h1_h2_h3_df.round(3)
h1_h2_h3_df['absolute_rho'] = np.abs(h1_h2_h3_df['Spearman $\rho$'])
h1_h2_h3_df.sort_values(by=['Dependent Variable', 'absolute_rho', 'Independent Variable'],
                        ascending=False, inplace=True)
h1_h2_h3_df.set_index(['Dependent Variable', 'Independent Variable'], inplace=True)
h1_h2_h3_df.drop(columns='absolute_rho', inplace=True)

# In [ ]:

h1_h2_h3_df[h1_h2_h3_df['Decision'] == 'Reject $H_0$'][['Spearman $\rho$', '$p$-value']]

# In [ ]:

h1_h2_h3_df[h1_h2_h3_df['Decision'] == 'Reject $H_0$'][['Spearman $\rho$', '$p$-value']].
to_csv('../Data/Tables/correlation_results.csv')
h1_h2_h3_df[h1_h2_h3_df['Decision'] == 'Reject $H_0$'][['Spearman $\rho$', '$p$-value']]

# ## Model Training

# In [ ]:

param_grid = {
    'n_estimators': [100, 200, 500],
    'max_depth': [10, 20, 30, None],
    'min_samples_split': [2, 5, 10, 20],
    'min_samples_leaf': [1, 2, 4, 10],
    'max_features': [1, 'sqrt', 'log2']
}

# In [ ]:

scores = []

# ### Gaussian Mixture Model Clustering

# In [ ]:

def evaluate_gmm(data, max_clusters=10):
    bics = []
    aics = []
    silhouette_scores = []

    # Scale the data
    scaler = StandardScaler()
    data_scaled = scaler.fit_transform(data)

    for n in range(2, max_clusters + 1):
        gmm = GaussianMixture(n_components=n, random_state=42)
        gmm.fit(data_scaled)

        # Calculate BIC and AIC
        bics.append(gmm.bic(data_scaled))
        aics.append(gmm.aic(data_scaled))

        # Calculate Silhouette Score
        labels = gmm.predict(data_scaled)
        silhouette_scores.append(silhouette_score(data_scaled, labels))

    # Plot the results
    plt.figure(figsize=(12, 3))

    plt.plot(range(2, max_clusters + 1), bics, 'o-', label='BIC', color='lightskyblue')
    plt.plot(range(2, max_clusters + 1), aics, 'o-', label='AIC', color='royalblue')

```

```

plt.xlabel('Number of Clusters')
plt.ylabel('Information Criterion')
plt.legend()

plt.savefig('../Data/Figures/gmm_clusters.pdf', bbox_inches='tight')

plt.show()

# Determine the optimal number of clusters based on the minimum BIC and maximum
Silhouette Score
optimal_clusters_bic = np.argmin(bics) + 2 # +2 because range starts at 2

print(f"Optimal number of clusters based on BIC: {optimal_clusters_bic}")

return optimal_clusters_bic

data = cleaned_aggregated_data[independent_vars] # Replace with your actual data
optimal_clusters_bic = evaluate_gmm(data)

# In [ ]:

# Standardize the data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(cleaned_aggregated_data[independent_vars])

gmm = GaussianMixture(n_components=6, random_state=42)
gmm.fit(X_scaled)
probabilities = gmm.predict_proba(X_scaled)

# Add cluster probabilities as new columns
probabilities_df = pd.DataFrame(probabilities, columns=[f'cluster_prob_{i}' for i in range
(probabilities.shape[1])], index=cleaned_aggregated_data.index)

cleaned_aggregated_data = pd.concat([cleaned_aggregated_data, probabilities_df], axis=1)

cluster_cols = list(probabilities_df.columns)

cleaned_aggregated_data['cluster_id'] = np.argmax(cleaned_aggregated_data[cluster_cols],
axis=1)

cluster_cols_with_id = cluster_cols + ['cluster_id']

# In [ ]:

cleaned_aggregated_data

# ### Data Preparation for Regression

# In [ ]:

def prepare_training_dataset(X, include_interactions=True):
    original_index = X.index
    if include_interactions:
        # Include the interaction effect features
        poly = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
        X_poly = poly.fit_transform(X)

        # Get the feature names
        feature_names = poly.get_feature_names_out(X.columns)

        # Standardize the features
        scaler = StandardScaler()
        X = scaler.fit_transform(X_poly)
    else:
        feature_names = X.columns

    # Standardize the features

```

```

    scaler = StandardScaler()
    X = scaler.fit_transform(X)

    # Convert X_scaled to a DataFrame with the correct column names
    X = pd.DataFrame(X, columns=feature_names, index=original_index)

    return X, feature_names

# In[ ]:

# Select only the 9 independent variables for training
X, feature_names = prepare_training_dataset(cleaned_aggregated_data[independent_vars],
    False)

# Select the 9 independent variables and the GMM cluster probabilities for training
X_gmm, gmm_feature_names = prepare_training_dataset(cleaned_aggregated_data[
    independent_vars + cluster_cols], False)

# Select the 9 independent variables, the GMM cluster probabilities, and the interaction
terms for training
X_int, int_feature_names = prepare_training_dataset(cleaned_aggregated_data[
    independent_vars], True)

# Select the 9 independent variables, the GMM cluster probabilities, and the interaction
terms for training
X_gmm_int, gmm_int_feature_names = prepare_training_dataset(cleaned_aggregated_data[
    independent_vars + cluster_cols], True)

# In[ ]:

# Select the target dependent variable
target_dependent_var = 'dominance'
target_aggregated_dependent_var = target_dependent_var + '_' + target_aggregation
y = cleaned_aggregated_data[target_aggregated_dependent_var]

# ## Linear model

# ##### Simple linear regression model

# In[ ]:

lin_model, selected_features, p_values, train_index, test_index, train_mse_scores,
    test_mse_scores, train_r2_scores, test_r2_scores, average_mse_scores,
    average_r2_scores = k_fold_training_linear_model(X, y)

# In[ ]:

# Split the data
X_train, X_test, y_train, y_test = X.iloc[train_index], X.iloc[test_index], y.iloc[
    train_index], y.iloc[test_index]

# In[ ]:

lin_model_df = pd.DataFrame({'Feature': lin_model.feature_names_in_, 'Coefficient':
    lin_model.coef_, '$p$-value': p_values.loc[selected_features].to_numpy().flatten()})
lin_model_df['Absolute Coefficient'] = np.abs(lin_model_df['Coefficient'])
lin_model_df.sort_values(by='Absolute Coefficient', ascending=False, inplace=True)
lin_model_df.set_index('Feature', inplace=True)
lin_model_df.index = lin_model_df.index.map(lambda feature: titles.get(feature, feature))
lin_model_df.drop(columns='Absolute Coefficient', inplace=True)
lin_model_df.head(10).round(3).to_csv(f'../Data/Tables/lin_model_{target_dependent_var}
    _coefficients.csv')

```



```

# In[ ]:

lin_model_df.head(10).round(3)

# In[ ]:

print_scores(train_mse_scores, test_mse_scores, train_r2_scores, test_r2_scores)

# In[ ]:

plot_performance(target_aggregated_dependent_var, lin_model, f'lin_model_{
    target_dependent_var}', X_train[selected_features], y_train, X_test[selected_features
    ], y_test)

# In[ ]:

plot_residuals(target_aggregated_dependent_var, lin_model, f'lin_model_{
    target_dependent_var}', X_train[selected_features], y_train, X_test[selected_features
    ], y_test)

# In[ ]:

scores.append({'model_type': 'lin_model', 'score_type': 'mse_scores', 'dependent_variable'
    : target_dependent_var, 'baseline_scores': average_mse_scores, 'scores':
    test_mse_scores, 'train_scores': train_mse_scores})
scores.append({'model_type': 'lin_model', 'score_type': 'r2_scores', 'dependent_variable':
    target_dependent_var, 'baseline_scores': average_r2_scores, 'scores': test_r2_scores,
    'train_scores': train_r2_scores})

# In[ ]:

save_model(lin_model, f'lin_model_{target_dependent_var}')

##### Linear regression model with interactions

# In[ ]:

lin_int_model, selected_features, p_values, train_index, test_index, train_mse_scores,
    int_test_mse_scores, train_r2_scores, int_test_r2_scores, average_mse_scores,
    average_r2_scores = k_fold_training_linear_model(X_int, y)

# In[ ]:

# Split the data
X_train, X_test, y_train, y_test = X_int.iloc[train_index], X_int.iloc[test_index], y.iloc
    [train_index], y.iloc[test_index]

# In[ ]:

lin_int_model_df = pd.DataFrame({'Feature': lin_int_model.feature_names_in_, 'Coefficient'
    : lin_int_model.coef_, '$p$-value': p_values.loc[selected_features].to_numpy().flatten
    ()})
lin_int_model_df['Absolute Coefficient'] = np.abs(lin_int_model_df['Coefficient'])
lin_int_model_df.sort_values(by='Absolute Coefficient', ascending=False, inplace=True)
lin_int_model_df.set_index('Feature', inplace=True)

```

```

lin_int_model_df.index = lin_int_model_df.index.map(lambda feature: titles.get(feature,
    feature))
lin_int_model_df.drop(columns='Absolute Coefficient', inplace=True)
lin_int_model_df.head(10).round(3).to_csv(f'../Data/Tables/lin_int_model_{
    target_dependent_var}_coefficients.csv')

# In[ ]:

lin_int_model_df.head(10).round(3)

# In[ ]:

print_scores(train_mse_scores, int_test_mse_scores, train_r2_scores, int_test_r2_scores)

# In[ ]:

plot_performance(target_aggregated_dependent_var, lin_int_model, f'lin_int_model_{
    target_dependent_var}', X_train[selected_features], y_train, X_test[selected_features
    ], y_test)

# In[ ]:

plot_residuals(target_aggregated_dependent_var, lin_int_model, f'lin_int_model_{
    target_dependent_var}', X_train[selected_features], y_train, X_test[selected_features
    ], y_test)

# In[ ]:

scores.append({'model_type': 'lin_int_model', 'score_type': 'mse_scores', '
    dependent_variable': target_dependent_var, 'baseline_scores': test_mse_scores, 'scores'
    ': int_test_mse_scores, 'train_scores': train_mse_scores})
scores.append({'model_type': 'lin_int_model', 'score_type': 'r2_scores', '
    dependent_variable': target_dependent_var, 'baseline_scores': test_r2_scores, 'scores'
    ': int_test_r2_scores, 'train_scores': train_r2_scores})

# In[ ]:

save_model(lin_int_model, f'lin_int_model_{target_dependent_var}')

# #### Linear regression model with GMM cluster probabilities

# In[ ]:

lin_gmm_model, selected_features, p_values, train_index, test_index, train_mse_scores,
    gmm_test_mse_scores, train_r2_scores, gmm_test_r2_scores, average_mse_scores,
    average_r2_scores = k_fold_training_linear_model(X_gmm, y)

# In[ ]:

# Split the data
X_train, X_test, y_train, y_test = X_gmm.iloc[train_index], X_gmm.iloc[test_index], y.iloc
    [train_index], y.iloc[test_index]

# In[ ]:

```

```

lin_gmm_model_df = pd.DataFrame({'Feature': lin_gmm_model.feature_names_in_, 'Coefficient'
    : lin_gmm_model.coef_, '$p$-value': p_values.loc[selected_features].to_numpy().flatten
    ()})
lin_gmm_model_df['Absolute Coefficient'] = np.abs(lin_gmm_model_df['Coefficient'])
lin_gmm_model_df.sort_values(by='Absolute Coefficient', ascending=False, inplace=True)
lin_gmm_model_df.set_index('Feature', inplace=True)
lin_gmm_model_df.index = lin_gmm_model_df.index.map(lambda feature: titles.get(feature,
    feature))
lin_gmm_model_df.drop(columns='Absolute Coefficient', inplace=True)
lin_gmm_model_df.head(10).round(3).to_csv(f'../Data/Tables/lin_gmm_model_{
    target_dependent_var}_coefficients.csv')

# In[ ]:

lin_gmm_model_df.head(10).round(3)

# In[ ]:

print_scores(train_mse_scores, gmm_test_mse_scores, train_r2_scores, gmm_test_r2_scores)

# In[ ]:

plot_performance(target_aggregated_dependent_var, lin_gmm_model, f'lin_gmm_model_{
    target_dependent_var}', X_train[selected_features], y_train, X_test[selected_features
    ], y_test)

# In[ ]:

plot_residuals(target_aggregated_dependent_var, lin_gmm_model, f'lin_gmm_model_{
    target_dependent_var}', X_train[selected_features], y_train, X_test[selected_features
    ], y_test)

# In[ ]:

scores.append({'model_type': 'lin_gmm_model', 'score_type': 'mse_scores', '
    dependent_variable': target_dependent_var, 'baseline_scores': test_mse_scores, 'scores
    ': gmm_test_mse_scores, 'train_scores': train_mse_scores})
scores.append({'model_type': 'lin_gmm_model', 'score_type': 'r2_scores', '
    dependent_variable': target_dependent_var, 'baseline_scores': test_r2_scores, 'scores'
    : gmm_test_r2_scores, 'train_scores': train_r2_scores})

# In[ ]:

save_model(lin_gmm_model, f'lin_gmm_model_{target_dependent_var}')

# #### Linear regression model with GMM cluster probabilities and interactions

# In[ ]:

lin_gmm_int_model, selected_features, p_values, train_index, test_index, train_mse_scores,
    gmm_int_test_mse_scores, train_r2_scores, gmm_int_test_r2_scores, average_mse_scores,
    average_r2_scores = k_fold_training_linear_model(X_gmm_int, y)

# In[ ]:

# Split the data

```

```

X_train, X_test, y_train, y_test = X_gmm_int.iloc[train_index], X_gmm_int.iloc[test_index
], y.iloc[train_index], y.iloc[test_index]

# In[ ]:

lin_gmm_int_model_df = pd.DataFrame({'Feature': lin_gmm_int_model.feature_names_in_, '
Coefficient': lin_gmm_int_model.coef_, '$p$-value': p_values.loc[selected_features].
to_numpy().flatten()})
lin_gmm_int_model_df['Absolute Coefficient'] = np.abs(lin_gmm_int_model_df['Coefficient'])
lin_gmm_int_model_df.sort_values(by='Absolute Coefficient', ascending=False, inplace=True)
lin_gmm_int_model_df.set_index('Feature', inplace=True)
lin_gmm_int_model_df.index = lin_gmm_int_model_df.index.map(lambda feature: titles.get(
feature, feature))
lin_gmm_int_model_df.drop(columns='Absolute Coefficient', inplace=True)
lin_gmm_int_model_df.head(10).round(3).to_csv(f'../Data/Tables/lin_gmm_int_model_{
target_dependent_var}_coefficients.csv')

# In[ ]:

lin_gmm_int_model_df.head(10).round(3)

# In[ ]:

print_scores(train_mse_scores, gmm_int_test_mse_scores, train_r2_scores,
gmm_int_test_r2_scores)

# In[ ]:

plot_performance(target_aggregated_dependent_var, lin_gmm_int_model, f'lin_gmm_int_model_{
target_dependent_var}', X_train[selected_features], y_train, X_test[selected_features
], y_test)

# In[ ]:

plot_residuals(target_aggregated_dependent_var, lin_gmm_int_model, f'lin_gmm_int_model_{
target_dependent_var}', X_train[selected_features], y_train, X_test[selected_features
], y_test)

# In[ ]:

scores.append({'model_type': 'lin_gmm_int_model', 'score_type': 'mse_scores', '
dependent_variable': target_dependent_var, 'baseline_scores': gmm_test_mse_scores, '
scores': gmm_int_test_mse_scores, 'train_scores': train_mse_scores})
scores.append({'model_type': 'lin_gmm_int_model', 'score_type': 'r2_scores', '
dependent_variable': target_dependent_var, 'baseline_scores': gmm_test_r2_scores, '
scores': gmm_int_test_r2_scores, 'train_scores': train_r2_scores})

# In[ ]:

save_model(lin_gmm_int_model, f'lin_gmm_int_model_{target_dependent_var}')

# ## Random forest model

# #### Simple random forest regression model

# In[ ]:

```

```

rf_model, train_index, test_index, train_mse_scores, test_mse_scores, train_r2_scores,
test_r2_scores, average_mse_scores, average_r2_scores =
    k_fold_training_random_forest_model(X.to_numpy(), y, param_grid)

# In[ ]:

# Split the data
X_train, X_test, y_train, y_test = X.iloc[train_index], X.iloc[test_index], y.iloc[
    train_index], y.iloc[test_index]

# In[ ]:

rf_model.get_params()

# In[ ]:

rf_model_df = pd.DataFrame({'Feature': feature_names, 'Importance': rf_model.
    feature_importances_})
rf_model_df.set_index('Feature', inplace=True)

# In[ ]:

plot_importances(rf_model_df, f'rf_model_{target_dependent_var}_importances')

# In[ ]:

print_scores(train_mse_scores, test_mse_scores, train_r2_scores, test_r2_scores)

# In[ ]:

plot_performance(target_aggregated_dependent_var, rf_model, f'rf_model_{
    target_dependent_var}', X_train.to_numpy(), y_train, X_test.to_numpy(), y_test)

# In[ ]:

plot_residuals(target_aggregated_dependent_var, rf_model, f'rf_model_{target_dependent_var
    }', X_train.to_numpy(), y_train, X_test.to_numpy(), y_test)

# In[ ]:

scores.append({'model_type': 'rf_model', 'score_type': 'mse_scores', 'dependent_variable':
    target_dependent_var, 'baseline_scores': average_mse_scores, 'scores':
    test_mse_scores, 'train_scores': train_mse_scores})
scores.append({'model_type': 'rf_model', 'score_type': 'r2_scores', 'dependent_variable':
    target_dependent_var, 'baseline_scores': average_r2_scores, 'scores': test_r2_scores,
    'train_scores': train_r2_scores})

# In[ ]:

save_model(rf_model, f'rf_model_{target_dependent_var}')

##### Random forest regression model with interactions

# In[ ]:

```

```

rf_int_model, train_index, test_index, train_mse_scores, int_test_mse_scores,
    train_r2_scores, int_test_r2_scores, average_mse_scores, average_r2_scores =
    k_fold_training_random_forest_model(X_int.to_numpy(), y, param_grid)

# In[ ]:

# Split the data
X_train, X_test, y_train, y_test = X_int.iloc[train_index], X_int.iloc[test_index], y.iloc
    [train_index], y.iloc[test_index]

# In[ ]:

rf_int_model.get_params()

# In[ ]:

rf_int_model_df = pd.DataFrame({'Feature': int_feature_names, 'Importance': rf_int_model.
    feature_importances_})
rf_int_model_df.set_index('Feature', inplace=True)

# In[ ]:

plot_importances(rf_int_model_df, f'rf_int_model_{target_dependent_var}_importances')

# In[ ]:

print_scores(train_mse_scores, int_test_mse_scores, train_r2_scores, int_test_r2_scores)

# In[ ]:

plot_performance(target_aggregated_dependent_var, rf_int_model, f'rf_int_model_{
    target_dependent_var}', X_train.to_numpy(), y_train, X_test.to_numpy(), y_test)

# In[ ]:

plot_residuals(target_aggregated_dependent_var, rf_int_model, f'rf_int_model_{
    target_dependent_var}', X_train.to_numpy(), y_train, X_test.to_numpy(), y_test)

# In[ ]:

scores.append({'model_type': 'rf_int_model', 'score_type': 'mse_scores', '
    dependent_variable': target_dependent_var, 'baseline_scores': test_mse_scores, 'scores
    ': int_test_mse_scores, 'train_scores': train_mse_scores})
scores.append({'model_type': 'rf_int_model', 'score_type': 'r2_scores', '
    dependent_variable': target_dependent_var, 'baseline_scores': test_r2_scores, 'scores'
    ': int_test_r2_scores, 'train_scores': train_r2_scores})

# In[ ]:

save_model(rf_int_model, f'rf_int_model_{target_dependent_var}')

# #### Random forest regression model with GMM cluster probabilities

```

```

# In[ ]:

rf_gmm_model, train_index, test_index, train_mse_scores, gmm_test_mse_scores,
    train_r2_scores, gmm_test_r2_scores, average_mse_scores, average_r2_scores =
    k_fold_training_random_forest_model(X_gmm.to_numpy(), y, param_grid)

# In[ ]:

# Split the data
X_train, X_test, y_train, y_test = X_gmm.iloc[train_index], X_gmm.iloc[test_index], y.iloc
    [train_index], y.iloc[test_index]

# In[ ]:

rf_gmm_model.get_params()

# In[ ]:

rf_gmm_model_df = pd.DataFrame({'Feature': gmm_feature_names, 'Importance': rf_gmm_model.
    feature_importances_})
rf_gmm_model_df.set_index('Feature', inplace=True)

# In[ ]:

plot_importances(rf_gmm_model_df, f'rf_gmm_model_{target_dependent_var}_importances')

# In[ ]:

print_scores(train_mse_scores, gmm_test_mse_scores, train_r2_scores, gmm_test_r2_scores)

# In[ ]:

plot_performance(target_aggregated_dependent_var, rf_gmm_model, f'rf_gmm_model_{
    target_dependent_var}', X_train.to_numpy(), y_train, X_test.to_numpy(), y_test)

# In[ ]:

plot_residuals(target_aggregated_dependent_var, rf_gmm_model, f'rf_gmm_model_{
    target_dependent_var}', X_train.to_numpy(), y_train, X_test.to_numpy(), y_test)

# In[ ]:

scores.append({'model_type': 'rf_gmm_model', 'score_type': 'mse_scores', '
    dependent_variable': target_dependent_var, 'baseline_scores': test_mse_scores, 'scores
    ': gmm_test_mse_scores, 'train_scores': train_mse_scores})
scores.append({'model_type': 'rf_gmm_model', 'score_type': 'r2_scores', '
    dependent_variable': target_dependent_var, 'baseline_scores': test_r2_scores, 'scores'
    ': gmm_test_r2_scores, 'train_scores': train_r2_scores})

# In[ ]:

save_model(rf_gmm_model, f'rf_gmm_model_{target_dependent_var}')

```

```
# #### Random forest regression model with GMM cluster probabilities and interactions
# In[ ]:

rf_gmm_int_model, train_index, test_index, train_mse_scores, gmm_int_test_mse_scores,
    train_r2_scores, gmm_int_test_r2_scores, average_mse_scores, average_r2_scores =
    k_fold_training_random_forest_model(X_gmm_int.to_numpy(), y, param_grid)

# In[ ]:

# Split the data
X_train, X_test, y_train, y_test = X_gmm_int.iloc[train_index], X_gmm_int.iloc[test_index
    ], y.iloc[train_index], y.iloc[test_index]

# In[ ]:

rf_gmm_int_model.get_params()

# In[ ]:

rf_gmm_int_model_df = pd.DataFrame({'Feature': gmm_int_feature_names, 'Importance':
    rf_gmm_int_model.feature_importances_})
rf_gmm_int_model_df.set_index('Feature', inplace=True)

# In[ ]:

plot_importances(rf_gmm_int_model_df, f'rf_gmm_int_model_{target_dependent_var}'
    _importances')

# In[ ]:

print_scores(train_mse_scores, gmm_int_test_mse_scores, train_r2_scores,
    gmm_int_test_r2_scores)

# In[ ]:

plot_performance(target_aggregated_dependent_var, rf_gmm_int_model, f'rf_gmm_int_model_{
    target_dependent_var}', X_train.to_numpy(), y_train, X_test.to_numpy(), y_test)

# In[ ]:

plot_residuals(target_aggregated_dependent_var, rf_gmm_int_model, f'rf_gmm_int_model_{
    target_dependent_var}', X_train.to_numpy(), y_train, X_test.to_numpy(), y_test)

# In[ ]:

scores.append({'model_type': 'rf_gmm_int_model', 'score_type': 'mse_scores', '
    dependent_variable': target_dependent_var, 'baseline_scores': gmm_test_mse_scores, '
    scores': gmm_int_test_mse_scores, 'train_scores': train_mse_scores})
scores.append({'model_type': 'rf_gmm_int_model', 'score_type': 'r2_scores', '
    dependent_variable': target_dependent_var, 'baseline_scores': gmm_test_r2_scores, '
    scores': gmm_int_test_r2_scores, 'train_scores': train_r2_scores})

# In[ ]:
```



```

save_model(rf_gmm_int_model, f'rf_gmm_int_model_{target_dependent_var}')

# ## Hypotheses Testing RQ4

# In[ ]:

scores_df = pd.DataFrame(scores)
scores_df.set_index(['score_type', 'dependent_variable', 'model_type'], inplace=True)
scores_df['overfit_ratio'] = scores_df['train_scores'] / scores_df['baseline_scores']

# In[ ]:

scores_df.to_csv(f'../Data/Tables/scores.csv')
scores_df

# In[ ]:

t_test_results_df = t_test(scores_df, score_types, model_types, dependent_vars, names=['
    scores', 'baseline_scores'])

# In[ ]:

t_test_results_df.to_csv(f'../Data/Tables/t_test_results.csv')
t_test_results_df

# In[ ]:

selected = []

# In[ ]:

selected.append(t_test_results_df.loc['mse_scores', :][['baseline_scores', 'scores', '
    t_statistic', 'p_negative']].rename(columns={'t_statistic': 'statistic', 'p_negative'
    : 'p_value'}))
selected.append(t_test_results_df.loc['r2_scores', :][['baseline_scores', 'scores', '
    t_statistic', 'p_positive']].rename(columns={'t_statistic': 'statistic', 'p_positive'
    : 'p_value'}))

# In[ ]:

h4_df = pd.concat(selected, axis=0)
h4_df.index = t_test_results_df.index
h4_df['train_scores'] = scores_df['train_scores']
h4_df[['baseline_scores', 'scores', 'train_scores']]

# In[ ]:

h4_df = pd.concat([h4_df[['train_scores', 'scores', 'baseline_scores']],
    bonferroni_correction(h4_df, alpha=0.05, apply_correction=False)], axis=1)
h4_df['baseline_scores'] = h4_df.baseline_scores.apply(np.mean)
h4_df['scores'] = h4_df.scores.apply(np.mean)
h4_df['train_scores'] = h4_df.train_scores.apply(np.mean)
h4_df.reset_index(inplace=True)
h4_df.sort_values(by=['model_type', 'score_type', 'dependent_variable'], inplace=True)

```

```

h4_df.dependent_variable = h4_df.dependent_variable.map(lambda feature: titles.get(feature
, feature))
h4_df.model_type = h4_df.model_type.map(lambda feature: titles.get(feature, feature))
h4_df.score_type = h4_df.score_type.map(lambda feature: titles.get(feature, feature))
h4_df.set_index(['model_type', 'score_type', 'dependent_variable'], inplace=True)
h4_df

# In [ ]:

h4_df = h4_df.round(3)
h4_df['reject_h0'] = h4_df['reject_h0'].astype("string")
h4_df.loc[h4_df['reject_h0'] == 'True', 'reject_h0'] = 'Reject $H_0$'
h4_df.loc[h4_df['reject_h0'] == 'False', 'reject_h0'] = 'Fail to reject $H_0$'
h4_df.reset_index(inplace=True)
h4_df.columns = ['Model Type', 'Score Type', 'Dependent Variable', "Training", 'Optimized'
, 'Baseline', '$t$-statistic', '$p$-value', 'Decision']

# In [ ]:

h4_df['Dataset'] = None
h4_df.loc[h4_df['Model Type'].str.contains('X'), 'Dataset'] = 'X'
h4_df.loc[h4_df['Model Type'].str.contains('X_int'), 'Dataset'] = 'X_int'
h4_df.loc[h4_df['Model Type'].str.contains('X_gmm'), 'Dataset'] = 'X_gmm'
h4_df.loc[h4_df['Model Type'].str.contains('X_gmm_int'), 'Dataset'] = 'X_gmm_int'
h4_df.loc[h4_df['Model Type'].str.contains('Linear'), 'Model Type'] = 'Linear'
h4_df.loc[h4_df['Model Type'].str.contains('Random forest'), 'Model Type'] = 'Random
Forest'
h4_df.sort_values(by=['Dependent Variable', 'Model Type', 'Dataset'], inplace=True)
h4_df.set_index(['Dependent Variable', 'Model Type', 'Dataset', 'Score Type'], inplace=
True)
h4_df.to_csv(f'../Data/Tables/h4_results.csv')

# In [ ]:

h4_df[h4_df.Decision == 'Reject $H_0$'].loc['Arousal', :, :, :][['Training', 'Optimized',
'Baseline', '$t$-statistic', '$p$-value']].to_csv(f'../Data/Tables/h4_results_arousal.
csv')
h4_df[h4_df.Decision == 'Reject $H_0$'].loc['Arousal', :, :, :][['Training', 'Optimized',
'Baseline', '$t$-statistic', '$p$-value']]

# In [ ]:

overfit_df = scores_df[['baseline_scores', 'scores', 'train_scores', 'overfit_ratio']].
copy()
overfit_df['ones'] = overfit_df.apply(lambda x: np.ones(5), axis=1)

# In [ ]:

t_test_overfitting_results_df = t_test(overfit_df, ['mse_scores'], model_types,
dependent_vars, names=['overfit_ratio', 'ones'])
t_test_overfitting_results_df.to_csv(f'../Data/Tables/t_test_overfitting_results.csv')
t_test_overfitting_results_df

# In [ ]:

selected = []
selected.append(t_test_overfitting_results_df[['baseline_scores', 'scores', 't_statistic',
'p_unequal',]].rename(columns={'t_statistic': 'statistic', 'p_unequal': 'p_value'}))

# In [ ]:

```

```

overfitting_df = pd.concat(selected, axis=0)
overfitting_df.index = t_test_overfitting_results_df.index
overfitting_df['train_scores'] = scores_df['train_scores']

# In[ ]:

overfitting_df

# In[ ]:

overfitting_df = pd.concat([overfitting_df[['scores', 'baseline_scores', 'train_scores']],
    bonferroni_correction(overfitting_df, alpha=0.05, apply_correction=False)], axis=1)
overfitting_df['scores'] = overfitting_df.scores.apply(np.mean)
overfitting_df['train_scores'] = overfitting_df.train_scores.apply(np.mean)
overfitting_df.reset_index(inplace=True)
overfitting_df.drop(columns=['score_type', 'baseline_scores'], inplace=True)
overfitting_df.sort_values(by=['dependent_variable', 'model_type'], inplace=True)
overfitting_df.dependent_variable = overfitting_df.dependent_variable.map(lambda feature:
    titles.get(feature, feature))
overfitting_df.model_type = overfitting_df.model_type.map(lambda feature: titles.get(
    feature, feature))
overfitting_df.set_index(['dependent_variable', 'model_type'], inplace=True)
overfitting_df = overfitting_df.round(3)
overfitting_df['reject_h0'] = overfitting_df['reject_h0'].astype("string")
overfitting_df.loc[overfitting_df['reject_h0'] == 'True', 'reject_h0'] = 'Reject $H_0$'
overfitting_df.loc[overfitting_df['reject_h0'] == 'False', 'reject_h0'] = 'Fail to reject
    $H_0$'
overfitting_df.reset_index(inplace=True)
overfitting_df

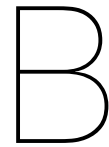
# In[ ]:

overfitting_df.columns = ['Dependent Variable', 'Model Type', '$OR$', 'Training MSE', '$t$
    -statistic', '$p$-value', 'Decision']
overfitting_df['Dataset'] = None
overfitting_df.loc[overfitting_df['Model Type'].str.contains('X'), 'Dataset'] = 'X'
overfitting_df.loc[overfitting_df['Model Type'].str.contains('X_int'), 'Dataset'] = 'X_int'
overfitting_df.loc[overfitting_df['Model Type'].str.contains('X_gmm'), 'Dataset'] = 'X_gmm'
overfitting_df.loc[overfitting_df['Model Type'].str.contains('X_gmm_int'), 'Dataset'] = '
    X_gmm_int'
overfitting_df.loc[overfitting_df['Model Type'].str.contains('Linear'), 'Model Type'] = '
    Linear'
overfitting_df.loc[overfitting_df['Model Type'].str.contains('Random forest'), 'Model Type
    '] = 'Random Forest'
overfitting_df.sort_values(by=['Dependent Variable', 'Model Type', 'Dataset'], inplace=
    True)
overfitting_df.set_index(['Dependent Variable', 'Model Type', 'Dataset'], inplace=True)
overfitting_df.to_csv(f'../Data/Tables/overfitting_results.csv')
overfitting_df

# In[ ]:

overfitting_df[overfitting_df.Decision == 'Fail to reject $H_0$'].loc['Dominance', :, :,
    :][['Training MSE', '$OR$', '$t$-statistic', '$p$-value']].to_csv(f'../Data/Tables/
    overfitting_results_dominance.csv')
overfitting_df[overfitting_df.Decision == 'Fail to reject $H_0$'].loc['Dominance', :, :,
    :][['Training MSE', '$OR$', '$t$-statistic', '$p$-value']]

```

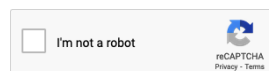



Survey

Designing Emotionally Expressive Behaviors for an Appearance-Constrained Robot

Start of Block: Captcha

Please verify that you are a human.



What is your Prolific ID?

Please note that this response should auto-fill with the correct ID.

End of Block: Captcha

Start of Block: Opening Statement and Informed Consent Form

Opening Statement

Dear Participant,

Welcome and thank you for choosing to be part of our research study.

Our objective is to understand how light, sound, and motion can make a faceless robot appear emotional. By watching videos of the robot and providing your feedback, you'll help us understand emotional communication by robots. Your contributions may be published in a scientific journal. This survey is expected to take about 25 minutes.

As a participant, you will:

- Watch videos of the robot in various situations.
- Evaluate the emotional expressions you perceive.
- Receive compensation upon survey completion.

Participant's Rights

- Your participation is voluntary, and you have the right to withdraw at any point.
- Even if you withdraw from the study, you will have 7 days to complete the survey in case that you change your mind (provided that there are still spots left in the survey quota).
- Compensation will be provided through Prolific.
- If you have any questions, please contact

Data Management

- No personally identifiable information will be collected to ensure your anonymity.
You can only participate in this survey once to maintain data integrity.
- Data will be securely and anonymously stored by Delft University of Technology, Leiden University, and 4TU.ResearchData, in the Netherlands.
- You will be asked to fill out an Informed Consent Form on the following page.

This study is led by:

- Fernando Corte Vargas, M.Sc. student at TU Delft, supervised by:
- Dr.-Ir. Jens Kober, Associate Professor at TU Delft
- Dr.-Ir. Joost Broekens, Associate Professor at Leiden University
- M.Sc. Bernhard Hilpert, PhD candidate at Leiden University

© 2024 Fernando Corte Vargas. All rights reserved.

Page Break

Informed Consent Form

I have read and understood the opening statement of this study.

Yes

No

I voluntarily consent to participate in this study and understand that I can withdraw at any time without giving a reason.

Yes

No

If I don't consent to participate in this study or decide to withdraw, I understand that I have up to 7 days to change my mind and re-enter the study, provided that there are still spots available in the survey quota.

Yes

No

I understand that I will be compensated through Prolific within 3 business days after I complete the survey satisfactorily.

Yes

No

I understand that completing the survey satisfactorily means that I will take enough time to carefully answer all questions in this survey.

Yes

No

I understand that the study will conclude on July 1, 2024.

Yes

No

I understand that all collected data is non-personal and anonymous.

Yes

No

I understand that collected data will be used for publication of a research paper after the study concludes.

Yes

No

I agree that my anonymous responses, views, or other input can be anonymously quoted in a research paper.

Yes

No

I give permission for the anonymous data I provide to be archived in the institutional repositories of Delft University of Technology, Leiden University, and 4TU.ResearchData for future research and learning.

Yes

No

I understand that access to these repositories is limited to non-commercial, research purposes only.

Yes

No

End of Block: Opening Statement and Informed Consent Form

Start of Block: Commitment Request

We care about the quality of our data, which is why it's important that you spend enough time to think about your answers to the questions of this survey.

To make sure of this, we have set up a minimum time requirement for some questions in the survey. This means that in some questions of the survey, you won't be able to proceed to the next question until a certain amount of time has passed.

Do you commit to spending enough time to provide thoughtful answers to the questions in this survey?

No, I will not

Yes, I will

End of Block: Commitment Request

Start of Block: Demographics

What country are you from?

▼ Afghanistan ... Zimbabwe

What is your age?

End of Block: Demographics

Start of Block: Example Questions Block

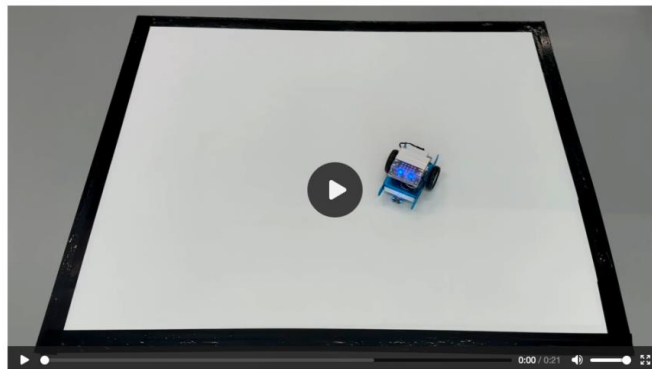
Video Snippets

Before you start the survey, we'd like to show you a few snippets of the videos that you might see in the survey. The purpose is to help you get an idea of how much the behaviors of the robot can vary across videos.

Please watch the video provided below. Click on the video to play it, and make sure you are using headphones or your computer's speakers to hear the audio. You cannot continue until the video finishes playing.

Feel free to replay the video as needed.

After you finish watching the video, you can continue to the next part of the survey.



Page Break

Example

To help you understand the questions in the survey, we provide an example with detailed explanations. Please read carefully to familiarize yourself with the questions we will ask.

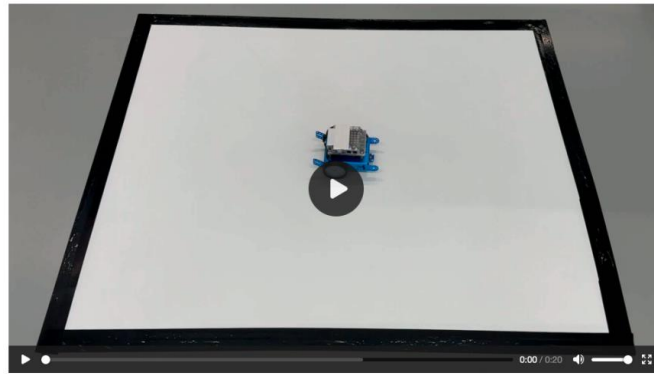
The upcoming survey questions will be identical to this one (without the detailed

explanations), **so please spend enough time to make sure you understand the questions before proceeding.**

Video

Please watch the video provided below. Click on the video to play it, and make sure you are using headphones or your computer's speakers to hear the audio. You cannot continue until the video finishes playing.

Feel free to replay the video as needed.



Questions

The following questions will measure your perception of the emotions displayed by the robot you just watched. Please read the instructions and answer the questions carefully.

Emotion Intensity Rating

Please rate the intensity of the emotion(s) as shown in the behavior of the robot.

Remember to give a rating for each emotion. If you didn't see a specific emotion, select "N/A". Please make sure you answer each row, otherwise, you won't be able to move forward in the survey.

If you saw an emotion not mentioned, please mention it in the "Other" section and rate its intensity.

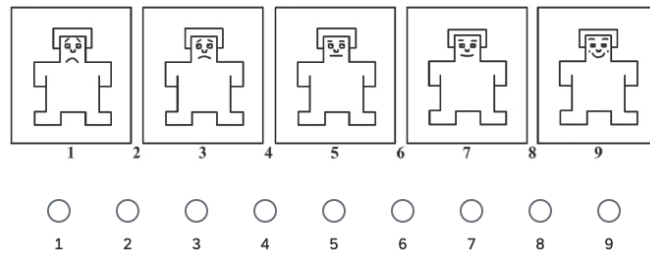
	N/A	Very Low	Low	Average	High	Very High
Joy	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sadness	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fear	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Anger	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Disgust	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Surprise	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Other	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

Dimensional Rating

Next, you'll rate **Pleasure**, which refers to the unpleasantness or pleasantness the robot seems to express through its behavior.

Look at the row of images below, which range from very negative to very positive.

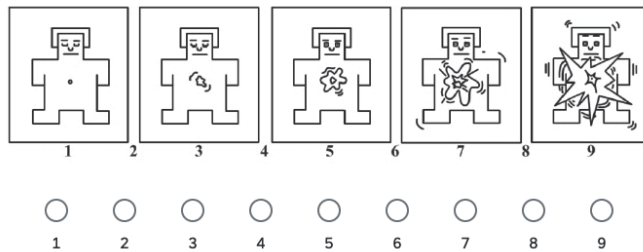
Use these images to choose a number to indicate how displeased or pleased the robot seemed to you.



Next, you'll rate **Arousal**, which indicates the alertness, awakens, and engagement the robot seems to express through its behavior.

Look at the images below, which range from very calm to very aroused.

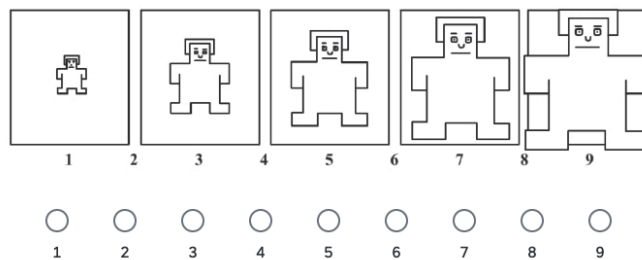
Use these images to select a number to indicate how calm or aroused the robot seemed to you.



Finally, you'll express thro

Look at the

Use them to



is to

seemed to

Appraisal Rating

Next, you will be asked to share your thoughts or analysis about what might cause the robot's emotional reactions in the video.

Please consider how the situation might affect (or has affected) its goals, desires, or overall well-being.

Below we provide a few examples:

- The robot looks like it's confused, maybe because it is figuring out what to do next. That is why it is moving slowly with dim lights.
- Maybe something unexpected happened to the robot, which leaves it seeming frustrated or afraid, moving fast and beeping loudly.
- The robot seems like it is maybe enjoying a peaceful moment, which leaves it relaxed and content, swaying gently and making calming sounds.

Please write an appraisal of the robot's emotions in the video.

Page Break

You've completed the example questions. By clicking "Next," you'll proceed to the survey where we'll start collecting your answers.

Please make sure you've read and understood the instructions provided in the example. That way, you will be more familiarized with the upcoming questions.

Please be aware that once you move forward, you won't be able to return to this section.

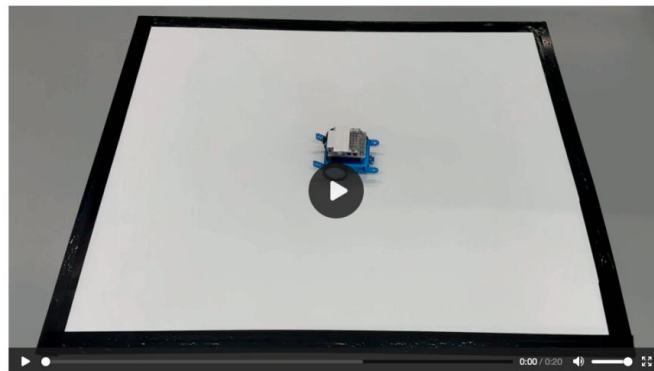
End of Block: Example Questions Block

Start of Block: Video 1

Video

Please watch the video provided below. Click on the video to play it, and make sure you are using headphones or your computer's speakers to hear the audio. You cannot continue until the video finishes playing.

Feel free to replay the video as needed.

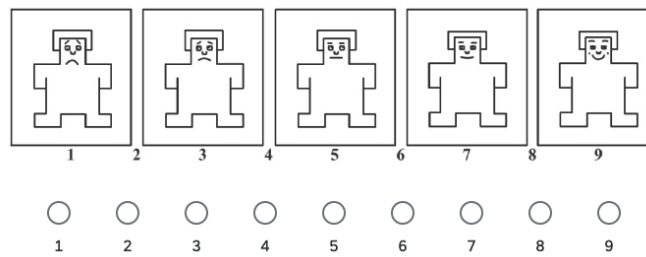


How intense were the emotions shown by the robot in the video? If you didn't see a specific emotion, select "N/A". Remember to give a rating for each emotion.

	N/A	Very Low	Low	Average	High	Very High
Joy	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sadness	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fear	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Anger	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Disgust	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Surprise	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Other	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

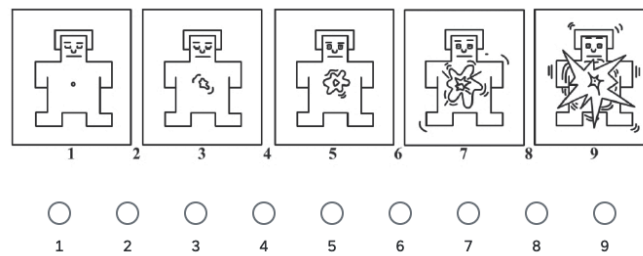
Please rate the pleasure of the robot (the unpleasantness or pleasantness the robot seems to express through its behavior).

Pleasure



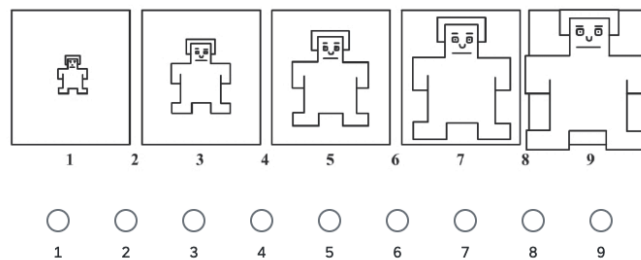
Please rate the arousal of the robot (the alertness, awakens, and engagement the robot seems to express through its behavior).

Arousal



Please rate the dominance of the robot (how much control the robot seems to express through its behavior).

Dominance



Please write your thoughts or analysis about what might cause the robot's emotional reactions in the video. Please consider how the situation might affect (or has affected) its goals, desires, or overall well-being.

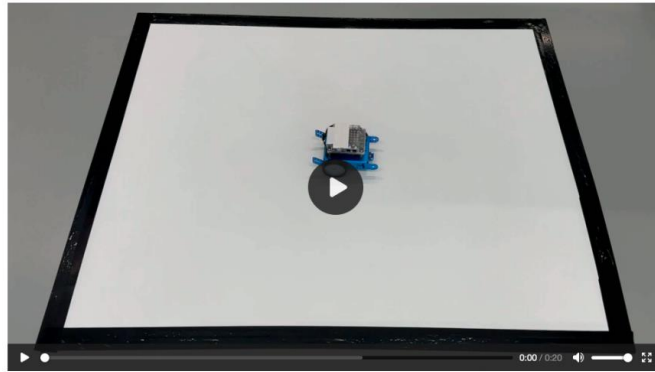
End of Block: Video 1

Start of Block: Video 2

Video

Please watch the video provided below. Click on the video to play it, and make sure you are using headphones or your computer's speakers to hear the audio. You cannot continue until the video finishes playing.

Feel free to replay the video as needed.

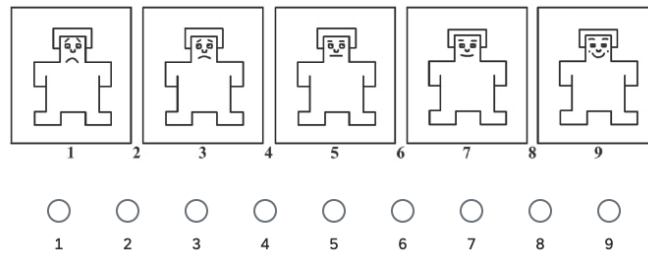


How intense were the emotions shown by the robot in the video? If you didn't see a specific emotion, select "N/A". Remember to give a rating for each emotion.

	N/A	Very Low	Low	Average	High	Very High
Joy	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sadness	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fear	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Anger	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Disgust	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Surprise	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Other	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

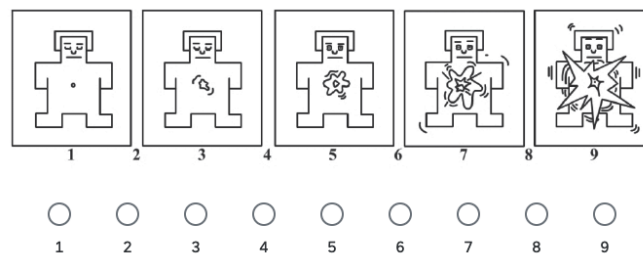
Please rate the pleasure of the robot (the unpleasantness or pleasantness the robot seems to express through its behavior).

Pleasure



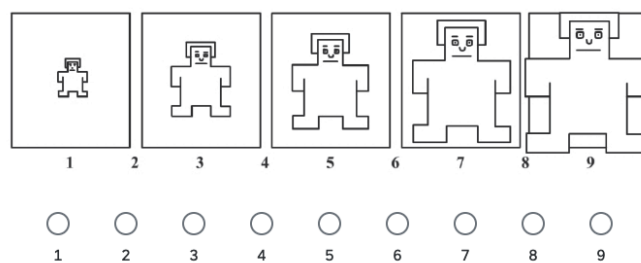
Please rate the arousal of the robot (the alertness, awakens, and engagement the robot seems to express through its behavior).

Arousal



Please rate the dominance of the robot (how much control the robot seems to express through its behavior).

Dominance



Please write your thoughts or analysis about what might cause the robot's emotional reactions in the video. Please consider how the situation might affect (or has affected) its goals, desires, or overall well-being.

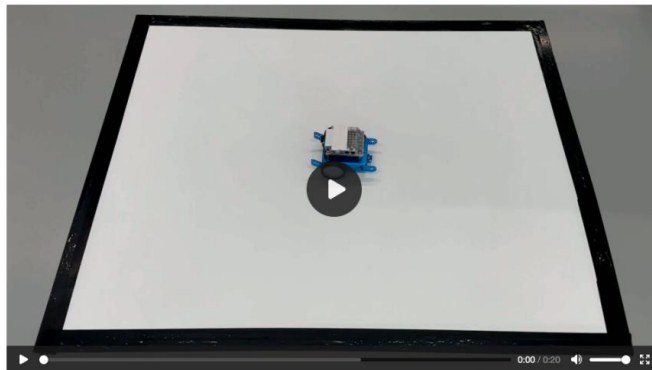
End of Block: Video 2

Start of Block: Video 3

Video

Please watch the video provided below. Click on the video to play it, and make sure you are using headphones or your computer's speakers to hear the audio. You cannot continue until the video finishes playing.

Feel free to replay the video as needed.

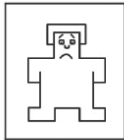
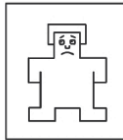
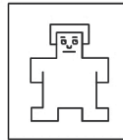
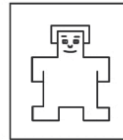
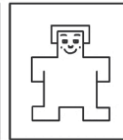


How intense were the emotions shown by the robot in the video? If you didn't see a specific emotion, select "N/A". Remember to give a rating for each emotion.

	N/A	Very Low	Low	Average	High	Very High
Joy	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sadness	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fear	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Anger	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Disgust	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Surprise	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Other	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

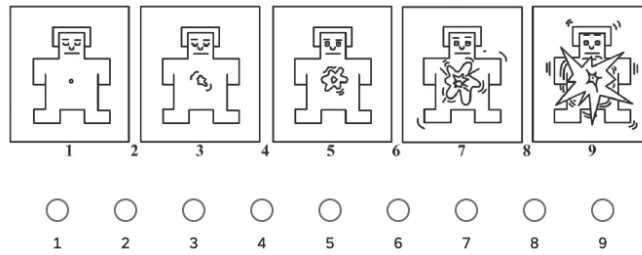
Please rate the pleasure of the robot (the unpleasantness or pleasantness the robot seems to express through its behavior).

Pleasure

								
1	2	3	4	5	6	7	8	9
<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
1	2	3	4	5	6	7	8	9

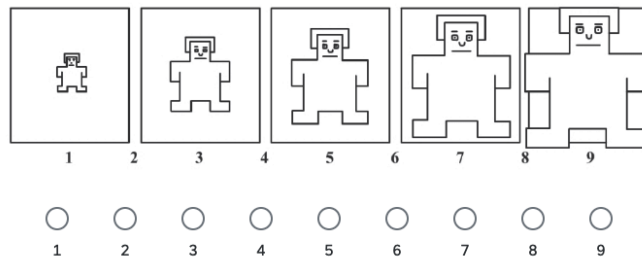
Please rate the arousal of the robot (the alertness, awakeness, and engagement the robot seems to express through its behavior).

Arousal



Please rate the dominance of the robot (how much control the robot seems to express through its behavior).

Dominance



Please write your thoughts or analysis about what might cause the robot's emotional reactions in the video. Please consider how the situation might affect (or has affected) its goals, desires, or overall well-being.

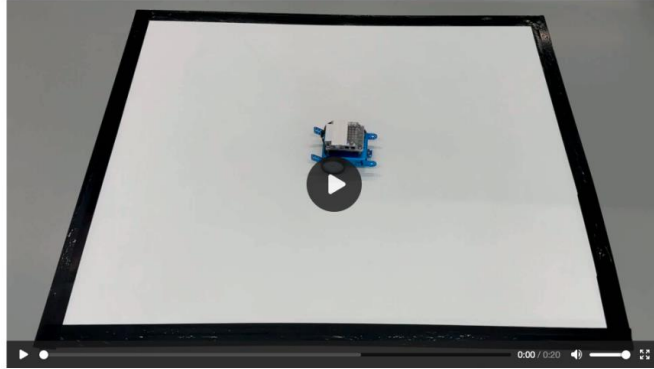
End of Block: Video 3

Start of Block: Video 4

Video

Please watch the video provided below. Click on the video to play it, and make sure you are using headphones or your computer's speakers to hear the audio. You cannot continue until the video finishes playing.

Feel free to replay the video as needed.

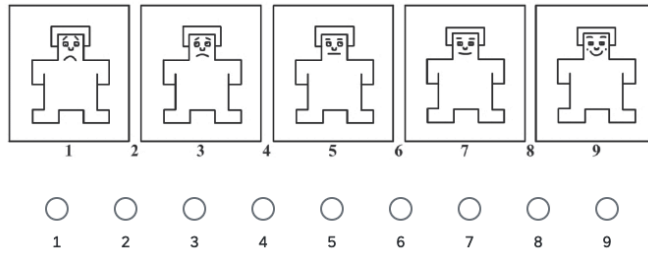


How intense were the emotions shown by the robot in the video? If you didn't see a specific emotion, select "N/A". Remember to give a rating for each emotion.

	N/A	Very Low	Low	Average	High	Very High
Joy	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Sadness	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Fear	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Anger	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Disgust	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Surprise	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>
Other	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>

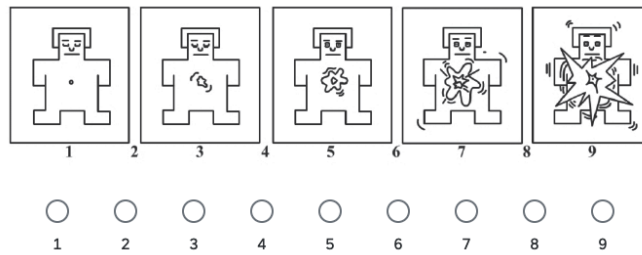
Please rate the pleasure of the robot (the unpleasantness or pleasantness the robot seems to express through its behavior).

Pleasure



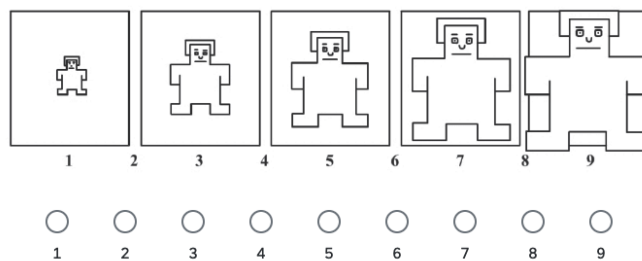
Please rate the arousal of the robot (the alertness, awakens, and engagement the robot seems to express through its behavior).

Arousal



Please rate the dominance of the robot (how much control the robot seems to express through its behavior).

Dominance



Please write your thoughts or analysis about what might cause the robot's emotional reactions in the video. Please consider how the situation might affect (or has affected) its goals, desires, or overall well-being.

End of Block: Video 4

Start of Block: End Of Survey

Congratulations!

Thank you for your participation! You have reached the end of the survey. Please click on "Next" to submit your answers.

Thank you for your interest in our study!

End of Block: End Of Survey

C

Human Research Ethics Approval Letter

Date 30-May-2024
Correspondence hrec@tudelft.nl



Human Research Ethics
Committee TU Delft
(<http://hrec.tudelft.nl>)

Visiting address
Jaffalaan 5 (building 31)
2628 BX Delft

Postal address
P.O. Box 5015 2600 GA Delft
The Netherlands

Ethics Approval Application: Designing Emotionally Expressive Behaviors for an Appearance-Constrained Robot
Applicant: Corte Vargas, Fernando

Dear Fernando Corte Vargas,

It is a pleasure to inform you that your application mentioned above has been approved.

Thanks very much for your submission to the HREC which has been approved.

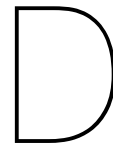
In addition to any specific conditions or notes, the HREC provides the following standard advice to all applicants:

- In light of recent tax changes, we advise that you confirm any proposed remuneration of research subjects with your faculty contract manager before going ahead.
- Please make sure when you carry out your research that you confirm contemporary covid protocols with your faculty HSE advisor, and that ongoing covid risks and precautions are flagged in the informed consent - with particular attention to this where there are physically vulnerable (eg: elderly or with underlying conditions) participants involved.
- Our default advice is not to publish transcripts or transcript summaries, but to retain these privately for specific purposes/checking; and if they are to be made public then only if fully anonymised and the transcript/summary itself approved by participants for specific purpose.
- Where there are collaborating (including funding) partners, appropriate formal agreements including clarity on responsibilities, including data ownership, responsibilities and access, should be in place and that relevant aspects of such agreements (such as access to raw or other data) are clear in the Informed Consent.

Good luck with your research!

Sincerely,

Dr. Ir. U. Pesch
Chair HREC
Faculty of Technology, Policy and Management



Risk Assessment

**Delft University of Technology
HUMAN RESEARCH ETHICS
CHECKLIST FOR HUMAN RESEARCH
(Version January 2022)**

IMPORTANT NOTES ON PREPARING THIS CHECKLIST

1. An HREC application should be submitted for every research study that involves human participants (as Research Subjects) carried out by TU Delft researchers
2. Your HREC application should be submitted and approved **before** potential participants are approached to take part in your study
3. All submissions from Master's Students for their research thesis need approval from the relevant Responsible Researcher
4. The Responsible Researcher must indicate their approval of the completeness and quality of the submission by signing and dating this form OR by providing approval to the corresponding researcher via email (included as a PDF with the full HREC submission)
5. There are various aspects of human research compliance which fall outside of the remit of the HREC, but which must be in place to obtain HREC approval. These often require input from internal or external experts such as [Faculty Data Stewards](#), [Faculty HSE advisors](#), the [TU Delft Privacy Team](#) or external [Medical research partners](#).
6. You can find detailed guidance on completing your HREC application [here](#)
7. Please note that incomplete submissions (whether in terms of documentation or the information provided therein) will be returned for completion **prior to any assessment**
8. If you have any feedback on any aspect of the HREC approval tools and/or process you can leave your comments [here](#)

I. Applicant Information

PROJECT TITLE:	Designing Emotionally Expressive Behaviors for an Appearance-Constrained Robot
Research period: <i>Over what period of time will this specific part of the research take place</i>	[TBD]
Faculty:	Mechanical Engineering
Department:	Cognitive Robotics
Type of the research project: <i>(Bachelor's, Master's, DreamTeam, PhD, PostDoc, Senior Researcher, Organisational etc.)</i>	Master's thesis
Funder of research: <i>(EU, NWO, TUD, other – in which case please elaborate)</i>	Leiden University
Name of Corresponding Researcher: <i>(If different from the Responsible Researcher)</i>	Joost Broekens
E-mail Corresponding Researcher: <i>(If different from the Responsible Researcher)</i>	Associate Professor
Position of Corresponding Researcher: <i>(Masters, DreamTeam, PhD, PostDoc, Assistant/ Associate/ Full Professor)</i>	d.j.broekens@liacs.leidenuniv.nl joost.broekens@gmail.com
Name of Responsible Researcher: <i>Note: all student work must have a named Responsible Researcher to approve, sign and submit this application</i>	Jens Kober
E-mail of Responsible Researcher: <i>Please ensure that an institutional email address (no Gmail, Yahoo, etc.) is used for all project documentation/ communications including Informed Consent materials</i>	j.kober@tudelft.nl
Position of Responsible Researcher : <i>(PhD, PostDoc, Associate/ Assistant/ Full Professor)</i>	Associate Professor

II. Research Overview

NOTE: You can find more guidance on completing this checklist [here](#)

a) Please summarise your research very briefly (100-200 words)

What are you looking into, who is involved, how many participants there will be, how they will be recruited and what are they expected to do?

Add your text here – (please avoid jargon and abbreviations)

The goal of this research is to investigate what features of motion, light, and sound in the behaviors of a non-humanoid, faceless robot may be perceived as emotional. To do this, a group of about 500 people will be recruited through Amazon's Mechanical Turk to respond to an online survey in which they will watch a series of videos of a robot executing certain behaviors, and will then rate any emotion qualities of the robot's behavior that may be perceived.

b) If your application is an additional project related to an existing approved HREC submission, please provide a brief explanation including the existing relevant HREC submission number/s.

Add your text here – (please avoid jargon and abbreviations)

--

-
- c) **If your application is a simple extension of, or amendment to,** an existing approved HREC submission, you can simply submit an [HREC Amendment Form](#) as a submission through LabServant.

III. Risk Assessment and Mitigation Plan

NOTE: You can find more guidance on completing this checklist [here](#)

Please complete the following table in full for all points to which your answer is “yes”. Bear in mind that the vast majority of projects involving human participants as Research Subjects also involve the collection of **Personally Identifiable Information (PII)** and/or **Personally Identifiable Research Data (PIRD)** which may pose potential risks to participants as detailed in Section G: Data Processing and Privacy below.

To ensure alignment between your risk assessment, data management and what you agree with your Research Subjects you can use the last two columns in the table below to refer to specific points in your Data Management Plan (DMP) and Informed Consent Form (ICF) – **but this is not compulsory**.

It’s worth noting that **you’re much more likely to need to resubmit your application if you neglect to identify potential risks**, than if you identify a potential risk and demonstrate how you will mitigate it. If necessary, the HREC will always work with you and colleagues in the Privacy Team and Data Management Services to see how, if at all possible, your research can be conducted.

ISSUE	Yes	No	RISK ASSESSMENT – what risks could arise? <i>Please ensure that you list ALL of the actual risks that could potentially arise – do not simply state whether you consider any such risks are important!</i>	MITIGATION PLAN – what mitigating steps will you take? <i>Please ensure that you summarise what actual mitigation measures you will take for each potential risk identified – do not simply state that you will e.g. comply with regulations.</i>	DMP <i>Please provide the relevant reference #</i>	ICF
<p>A. Partners and collaboration</p> <p>1. Will the research be carried out in collaboration with additional organisational partners such as:</p> <ul style="list-style-type: none"> One or more collaborating research and/or commercial organisations Either a research, or a work experience internship provider¹ <p>¹ <i>If yes, please include the graduation agreement in this application</i></p>	✓		<p>1. Leiden University and TU Delft may have different ethical standards and processes for approving research involving human subjects, leading to potential conflicts or misunderstandings about the ethical conduct of the research.</p> <p>2. The sharing of sensitive or personal data between institutions can lead to risks related to data privacy and security.</p> <p>3. Disagreements or misunderstandings about intellectual property rights, authorship, and publication rights might arise, potentially leading to conflicts.</p> <p>4. Collaborative projects can suffer from poor coordination and communication, leading to</p>	<p>1. No personal data is collected. There is no risk of personal data being published. Furthermore, the approval process is similar between Leiden University and TU Delft.</p> <p>2. No personal data is collected. Additionally, we will obtain informed consent from participants that clearly explains how their data will be used, stored, and shared between institutions.</p> <p>3. Intellectual property is shared between TU Delft and Leiden</p> <p>4. A robust project management framework has been implemented, including regular meetings, shared project management tools, and clear communication channels.</p> <p>5. We agree that there is no financial burden across the two universities, there is no money</p>	VI:33	05, ICF 06, 07, 08, 09, Q10, Q10

ISSUE	Yes	No	RISK ASSESSMENT – what risks could arise? Please ensure that you list ALL of the actual risks that could potentially arise – do not simply state whether you consider any such risks are important!	MITIGATION PLAN – what mitigating steps will you take? Please ensure that you summarise what actual mitigation measures you will take for each potential risk identified – do not simply state that you will e.g. comply with regulations.	DMP reference #	ICF
2. Is this research dependent on a Data Transfer or Processing Agreement with a collaborating partner or third party supplier? <i>If yes please provide a copy of the signed DTA/DPA</i>		✓				
3. Has this research been approved by another (externally) research ethics committee (e.g.: HREC and/or MREC/METC)? <i>If yes, please provide a copy of the approval (if possible) and summarise any key points in your Risk Management section below</i>		✓				
B: Location						
4. Will the research take place in a country or countries, other than the Netherlands, within the EU?		✓				
5. Will the research take place in a country or countries outside the EU?		✓				
6. Will the research take place in a place/region or of higher risk – including known dangerous locations (in any country) or locations with non-democratic regimes?		✓				
C: Participants						
7. Will the study involve participants who may be vulnerable and possibly (legally) unable to give informed consent? (e.g., children below the legal age for giving consent, people with learning difficulties, people living in care or nursing homes,)		✓				
8. Will the study involve participants who may be vulnerable under specific circumstances and in specific contexts, such as victims and witnesses of violence, including domestic violence; sex workers; members of minority groups, refugees, irregular migrants or dissidents?		✓				
9. Are the participants, outside the context of the research, in a dependent or subordinate position to the investigator (such as own children, own students or employees of either TU Delft and/or a collaborating partner organisation)?		✓				

If YES please complete the Risk Assessment and Mitigation Plan columns below.

RISK ASSESSMENT – what risks could arise?
Please ensure that you list ALL of the actual risks that could potentially arise – do not simply state whether you consider any such risks are important!

5. Delays, inefficiencies, or inconsistencies in research execution.
Misunderstandings or conflicts may arise regarding the allocation of funds and resources between the collaborating universities, affecting the project's progress and outcomes.

MITIGATION PLAN – what mitigating steps will you take?
Please ensure that you summarise what actual mitigation measures you will take for each potential risk identified – do not simply state that you will e.g. comply with regulations.

involved in the supervision of the master project.

Please provide the relevant reference #

ISSUE	Yes	No	If YES please complete the Risk Assessment and Mitigation Plan columns below.	Please provide the relevant reference #	DMP	ICF
<p><i>It is essential that you safeguard against possible adverse consequences of this situation (such as allowing a student's failure to participate to your satisfaction to affect your evaluation of their coursework).</i></p>						
<p>10. Is there a high possibility of re-identification for your participants? (e.g., do they have a very specialist job of which there are only a small number in a given country, are they members of a small community, or employees from a partner company collaborating in the research? Or are they one of only a handful of (expert) participants in the study?</p>		✓				
D: Recruiting Participants						
<p>11. Will your participants be recruited through your own, professional, channels such as conference attendance lists, or through specific networks/such as self-help groups</p>		✓				
<p>12. Will the participants be recruited or accessed in the longer term by a (legal or customary) gatekeeper? (e.g., an adult professional working with children; a community leader or family member who has this customary role – within or outside the EU; the data producer of a long-term cohort study)</p>		✓				
<p>13. Will you be recruiting your participants through a crowd-sourcing service and/or involve a third party data-gathering service, such as a survey platform?</p>	✓		<p>1. There's a risk of receiving responses from bots or participants misrepresenting themselves to quality for the study, which can compromise the authenticity and representativeness of the sample.</p> <p>2. Data collected via MTurk can vary in quality, with issues like rushed responses or participants not taking the study seriously, affecting the reliability of your results.</p> <p>3. Using third-party platforms for data collection and recruitment can lead to privacy concerns and potential data breaches, compromising participant data security.</p> <p>4. Ensuring that participants recruited through MTurk fully understand the study and provide informed consent can be challenging, especially when not interacting face-to-face.</p> <p>5. Relying on third-party platforms for critical aspects of your research makes the study</p>	<p>1. We will make use of Amazon Mechanical Turk's (MTurk) participant screening filters (called MTurk Masters) to verify the authenticity and quality of participants. This ensures that clearly eligible criteria are defined and enforced.</p> <p>2. We will recruit MTurk Masters to ensure quality data. Additionally, we can include attention check questions throughout the survey to identify and exclude participants not paying attention. We will offer fair compensation to encourage serious participation.</p> <p>3. Qualtrics complies with relevant data protection regulations and offers adequate data encryption and anonymization capabilities. We will collect only the data necessary, and ensure participant data anonymization is turned on to protect their identity.</p> <p>4. A clear and concise informed consent form is included that participants must agree to before participating in the study. This form includes</p>		<p>05, ICF 06, 07, 08, 09, Q10, Q10</p>

		If YES please complete the Risk Assessment and Mitigation Plan columns below.	Please provide the relevant reference #	ICF
ISSUE	Yes	No	RISK ASSESSMENT – what risks could arise? Please ensure that you list ALL of the actual risks that could potentially arise – do not simply state whether you consider any such risks are important!	MITIGATION PLAN – what mitigating steps will you take? Please ensure that you summarise what actual mitigation measures you will take for each potential risk identified – do not simply state that you will e.g. comply with regulations.
		vulnerable to changes in their policies, outages, or discontinuation of service.	<ol style="list-style-type: none"> 1. We will set compensation at a level that is fair and reflects the time and effort required but is not so high as to be the primary motivation for participation. This balance can help ensure that participants are motivated by interest in the study as well as financial compensation. 2. We will implement attention checks and minimum time requirements for completing the survey to discourage rushed responses. Additionally, we will use pilot testing to establish a reasonable completion time and set minimum time requirement based on these findings. 3. We will use Qualtrics's native capabilities to limit participation to one per individual, which consists of tracking unique identifiers (while maintaining privacy and compliance with data protection laws). It is made clear in the consent form that multiple submissions by the same person are not allowed and may result in disqualification from the study and forfeiture of compensation. 	
14. Will you be offering any financial, or other, remuneration to participants, and might this induce or bias participation?	✓	<ol style="list-style-type: none"> 1. Financial incentives might attract participants who are more interested in the compensation than the study itself, leading to a sample that does not accurately represent your target population. 2. Participants may rush through the survey to complete as many tasks as possible in a short time, compromising the quality of your data. 3. Participants may attempt to participate multiple times under different accounts to receive additional compensation. 4. The promise of financial remuneration might be seen as coercive, particularly if participants want to participate in the study to receive needed funds because of a possible precarious economic situation. 5. Financially motivated participation might lead to a sample that is not representative of the broader population, affecting the generalizability of our findings. 		

ISSUE	Yes	No	RISK ASSESSMENT – what risks could arise? Please ensure that you list ALL of the actual risks that could potentially arise – do not simply state whether you consider any such risks are important!	MITIGATION PLAN – what mitigating steps will you take? Please ensure that you summarize what actual mitigation measures you will take for each potential risk identified – do not simply state that you will e.g. comply with regulations.	Please provide the relevant reference #
<p>E: Subject Matter Research related to medical questions/health may require special attention. See also the website of the CCMO before contacting the HREC.</p> <p>15. Will your research involve any of the following:</p> <ul style="list-style-type: none"> • Medical research and/or clinical trials • Invasive sampling and/or medical imaging • Medical and <i>In Vitro</i> Diagnostic Medical/Devices Research <p>16. Will drugs, placebos, or other substances (e.g., drinks, foods, food or drink constituents, dietary supplements) be administered to the study participants? <i>If yes see here to determine whether medical ethical approval is required</i></p> <p>17. Will blood or tissue samples be obtained from participants? <i>If yes see here to determine whether medical ethical approval is required</i></p> <p>18. Does the study risk causing psychological stress or anxiety beyond that normally encountered by the participants in their life outside research? <i>If yes see here to determine whether medical ethical approval is required</i></p> <p>19. Will the study involve discussion of personal sensitive data which could put participants at increased legal, financial, reputational, security or other risk? (e.g., financial data, location data, data relating to children or other vulnerable groups) <i>Definitions of sensitive personal data, and special cases are provided on the TUD Privacy Team website.</i></p>			<p><i>If YES please complete the Risk Assessment and Mitigation Plan columns below.</i></p>	<p>4. We will assess the appropriateness of the compensation based on the expected demographics of the participants and the nature of the tasks involved. Ensure that the compensation is neither too low (which could be exploitative) nor too high (which could be coercive). We provide clear information in the consent form that participation is voluntary and that declining to participate will not penalize the participant in any way.</p> <p>5. We will use MITurk's native stratified sampling techniques to ensure the sample is representative of the population of interest. The target demographic must be clearly defined and we will use MITurk's screening to select participants who meet our criteria.</p>	<p>DMP</p> <p>ICF</p>

ISSUE	Yes	No	RISK ASSESSMENT – what risks could arise? <i>Please ensure that you list ALL of the actual risks that could potentially arise – do not simply state whether you consider any such risks are important!</i>	MITIGATION PLAN – what mitigating steps will you take? <i>Please ensure that you summarise what actual mitigation measures you will take for each potential risk identified – do not simply state that you will e.g. comply with regulations.</i>	DMP reference #	ICF
20. Will the study involve disclosing commercially or professionally sensitive, or confidential information? (e.g., relating to decision-making processes or business strategies which might, for example, be of interest to competitors)		✓				
21. Has your study been identified by the TU Delft Privacy Team as requiring a Data Processing Impact Assessment (DPIA)? <i>If yes please attach the advice/approval from the Privacy Team to this application.</i>		✓				
22. Does your research investigate causes or areas of conflict? <i>If yes please confirm that your fieldwork has been discussed with the appropriate safety/security advisors and approved by your Department/Faculty.</i>		✓				
23. Does your research involve observing illegal activities or data processed or provided by authorities responsible for preventing, investigating, detecting or prosecuting criminal offences <i>If so please confirm that your work has been discussed with the appropriate legal advisors and approved by your Department/Faculty.</i>		✓				
F: Research Methods						
24. Will it be necessary for participants to take part in the study without their knowledge and consent at the time? (e.g., covert observation of people in non-public places).		✓				
25. Will the study involve actively deceiving the participants? (For example, will participants be deliberately falsely informed, will information be withheld from them or will they be misled in such a way that they are likely to object or show unease when debriefed about the study).		✓				
26. Is pain or more than mild discomfort likely to result from the study? And/or could your research activity cause an accident involving (non-) participants?		✓				
27. Will the experiment involve the use of devices that are not CE certified? <i>Only, if yes; continue with the following questions:</i>		✓				
<ul style="list-style-type: none"> Was the device built in-house? Was it inspected by a safety expert at TU Delft? 						
<ul style="list-style-type: none"> <i>If yes, please provide a signed device report</i> If it was not built in-house and not CE-certified, was it inspected by some other, qualified authority in safety and approved? 						
28. Will your research involve face-to-face encounters with your participants and if so how will you assess and address Covid considerations?		✓				

If YES please complete the Risk Assessment and Mitigation Plan columns below.

Please provide the relevant reference #

ISSUE	Yes	No	<i>If YES please complete the Risk Assessment and Mitigation Plan columns below.</i> RISK ASSESSMENT – what risks could arise? <i>Please ensure that you list ALL of the actual risks that could potentially arise – do not simply state whether you consider any such risks are important!</i>	MITIGATION PLAN – what mitigating steps will you take? <i>Please ensure that you summarise what actual mitigation measures you will take for each potential risk identified – do not simply state that you will e.g. comply with regulations.</i>	Please provide the relevant reference # DMP	Please provide the relevant reference # ICF
29. Will your research involve either: a) "big data", combined datasets, new data-gathering or new data-merging techniques which might lead to re-identification of your participants and/or b) artificial intelligence or algorithm training where, for example biased datasets could lead to biased outcomes?		✓				
G: Data Processing and Privacy 30. Will the research involve collecting, processing and/or storing any directly identifiable PII (Personally Identifiable Information) including name or email address that will be used for administrative purposes only? (eg. obtaining Informed Consent or disbursing remuneration)		✓				
31. Will the research involve collecting, processing and/or storing any directly or indirectly identifiable PIRD (Personally Identifiable Research Data) including videos, pictures, IP address, gender, age etc and what other Personal Research Data (including personal or professional views) will you be collecting?		✓				
32. Will this research involve collecting data from the internet, social media and/or publicly available datasets which have been originally contributed by human participants		✓				
33. Will your research findings be published in one or more forms in the public domain, as e.g., Masters thesis, journal publication, conference presentation or wider public dissemination?	✓		<ol style="list-style-type: none"> 1. Publishing data could inadvertently reveal identifiable information about participants, leading to a breach of confidentiality. 2. Readers may misinterpret the findings, especially if complex data is not clearly presented or explained. 3. Publishing the research could expose the researchers to intellectual property theft, where others use the findings without proper attribution. 4. Expectations or requirements for data storing and sharing could compromise participant privacy or expose proprietary data. Additionally, there might be pressure to ensure that the findings are reproducible. 	<ol style="list-style-type: none"> 1. No personal data is collected. Further, data may be anonymized by removing or altering any identifiable information. Pseudonyms will be used and no demographic information that could be used to identify participants will be published. 2. Clarity and simplicity in the presentation of the findings must be ensured, both in the thesis and any journal publications. Limitations of the study will be included to guide interpretation. 3. Copyright notices will be used and publishing will be done in journals that allow retaining copyright over our work. 4. Data will be stored and shared in a way that is consistent with the guidelines established in the DMP and in the ICF. 	V, VI	O5, ICF, O6, O7, O8, O9, Q10, Q11
34. Will your research data be archived for re-use and/or teaching in an open, private or semi-open archive?	✓		<ol style="list-style-type: none"> 1. Sensitive data could be accessed and used by unauthorized individuals, leading to privacy 	<ol style="list-style-type: none"> 1. No personal data is collected. Furthermore, the principal investigators will implement strict access controls to ensure that data can only be 		

		If YES please complete the Risk Assessment and Mitigation Plan columns below.	Please provide the relevant reference #	ICF
ISSUE	Yes	No	RISK ASSESSMENT – what risks could arise? Please ensure that you list ALL of the actual risks that could potentially arise – do not simply state whether you consider any such risks are important!	MITIGATION PLAN – what mitigating steps will you take? Please ensure that you summarise what actual risk identified – do not simply state that you will e.g. comply with regulations.
			<p>violations and potential harm to research participants.</p> <ol style="list-style-type: none"> The risk that data could be altered, corrupted, or lost over time due to technical failures or inadequate data management practices. Failing to comply with legal, ethical, and regulatory standards related to data sharing, especially concerning personal data protection laws. Even with anonymized datasets, there is a risk of re-identification, especially with large or detailed datasets. Long-term sustainability of the archive could be jeopardized by funding cuts, technological obsolescence, or institutional changes. 	<p>accessed by authorized individuals. Authorization criteria are limited to non-commercial only.</p> <ol style="list-style-type: none"> We set out robust data management practices in the DMP thanks to the project storage of the TU Delft, which include regular backups, checks for integrity verification, and durable storage. We have reviewed and adhere to applicable laws and regulations (GDPR) in both the data collection and archiving phases. We have also ensured that the participant consent forms include permissions for data sharing and archiving in the manner planned. We will use Qualtrics's advanced anonymization techniques to ensure that re-identification of the participants' data is not possible. We will plan for long-term data storage supplied by Leiden University's funding and data storage technical infrastructure.

H: More on Informed Consent and Data Management

NOTE: You can find guidance and templates for preparing your Informed Consent materials) [here](#)

Your research involves human participants as Research Subjects if you are recruiting them or actively involving or influencing, manipulating or directing them in any way in your research activities. This means you must seek informed consent and agree/ implement appropriate safeguards regardless of whether you are collecting any PIRD.

Where you are also collecting PIRD, and using Informed Consent as the legal basis for your research, you need to also make sure that your IC materials are clear on any related risks and the mitigating measures you will take – including through responsible data management.

Got a comment on this checklist or the HREC process? You can leave your comments [here](#)

IV. Signature/s

Please note that by signing this checklist list as the sole, or Responsible, researcher you are providing approval of the completeness and quality of the submission, as well as confirming alignment between GDPR, Data Management and Informed Consent requirements.

Name of Corresponding Researcher (if different from the Responsible Researcher) (print)

Signature of Corresponding Researcher:

Date:

Name of Responsible Researcher (print)

Signature (or upload consent by mail) Responsible Researcher:

Date:

V. Completing your HREC application

Please use the following list to check that you have provided all relevant documentation

Required:

- **Always:** This completed HREC checklist
- **Always:** A data management plan (reviewed, where necessary, by a data-steward)
- **Usually:** A complete Informed Consent form (including Participant Information) and/or Opening Statement (for online consent)

Please also attach any of the following, if relevant to your research:

Document or approval	Contact/s
Full Research Ethics Application	After the assessment of your initial application HREC will let you know if and when you need to submit additional information
Signed, valid Device Report	Your Faculty HSE advisor
Ethics approval from an external Medical Committee	TU Delft Policy Advisor, Medical (Devices) Research
Ethics approval from an external Research Ethics Committee	Please append, if possible, with your submission
Approved Data Transfer or Data Processing Agreement	Your Faculty Data Steward and/or TU Delft Privacy Team
Approved Graduation Agreement	Your Master's thesis supervisor
Data Processing Impact Assessment (DPIA)	TU Delft Privacy Team
Other specific requirement	Please reference/explain in your checklist and append with your submission



Data Management Plan

Plan Overview

A Data Management Plan created using DMPonline

Title: Master's Thesis: Designing Emotionally Expressive Behaviors for an Appearance-Constrained Robot

Creator: Fernando Corte Vargas

Principal Investigator: Joost Broekens, Jens Kober

Data Manager: Joost Broekens, Jens Kober, Bernhard Hilpert

Affiliation: Delft University of Technology

Template: TU Delft Data Management Plan template (2021)

ORCID iD: 0000-0001-9198-898X

ORCID iD: 0000-0001-7257-5434

Project abstract:

The goal of this research is to investigate what features of motion, light, and sound in the behaviors of a non-humanoid, faceless robot may be perceived as emotional. To do this, a group of about 500 people will be recruited through Amazon's Mechanical Turk to respond to an online survey in which they will watch a series of videos of a robot executing certain behaviors, and will then rate any emotion qualities of the robot's behavior that may be perceived.

ID: 144484

Start date: 01-02-2024

Last modified: 10-04-2024

Master's Thesis: Designing Emotionally Expressive Behaviors for an Appearance-Constrained Robot

0. Administrative questions

1. Name of data management support staff consulted during the preparation of this plan.

My faculty data steward, Sara Shoghi, has reviewed this DMP on 03-04-2024

2. Date of consultation with support staff.

2024-03-11

I. Data description and collection or re-use of existing data

3. Provide a general description of the type of data you will be working with, including any re-used data:

Data

Type of data	File format(s)	How will data be collected (for re-used data: source and terms of use)?	Purpose of processing	Storage location	Who will have access to the data
Anonymized data on ratings of emotional perception from human participants of an appearance-constrained robot	.csv files	Online survey on Qualtrics	To understand any possible patterns in the recognition of emotional qualities of the behaviors of an appearance-constrained robot based on the modalities of light, motion and sound	TU Delft Project Data Storage (U:)	All three supervisors and myself
Informed consent form data	.csv files	Online survey on Qualtrics	This data will be collected prior to the beginning of the research survey. In order to be able to participate, each participant must give consent according to this form	TU Delft Project Data Storage (U:)	All three supervisors and myself
Videos of the appearance constrained robot that will be used for the online survey	.mp4 files	Video recordings	These videos will be used for the online survey, and will be stored for future research as well.	TU Delft Project Data Storage (U:)	All three supervisors and myself
Data with the input parameters that were used to generate each of the videos of the appearance constrained robot that will be used for the online survey	.csv files	Python script	This data is generated using a script that generates Sobol numerical sequences that are translated into the input parameters for the videos.	TU Delft Project Data Storage (U:)	All three supervisors and myself

4. How much data storage will you require during the project lifetime?

- 250 GB - 5 TB

Since the data will consist of a single .csv file, and no more than 1024 20-second long videos, the required data storage is not expected to surmount more than 500 GB.

II. Documentation and data quality

5. What documentation will accompany data?

- Data dictionary explaining the variables used
- Data will be deposited in a data repository at the end of the project (see section V) and data discoverability and re-usability will be ensured by adhering to the repository's metadata standards
- Methodology of data collection
- README file or other documentation explaining how data is organised

A README file will be created to make sure that the structure of the data is explained, as well as to clarify how the data can be used. Access to the dataset will be granted to researchers who want to use it, upon inspection and approval by one of the principal researchers.

Additionally, a data dictionary will be created for both the anonymized ratings from the participants (survey data), as well as for the data describing the input parameters of the videos of the robot.

III. Storage and backup during research process

6. Where will the data (and code, if applicable) be stored and backed-up during the project lifetime?

- Project Storage at TU Delft

IV. Legal and ethical requirements, codes of conduct

7. Does your research involve human subjects or 3rd party datasets collected from human participants?

- Yes

8A. Will you work with personal data? (information about an identified or identifiable natural person)

If you are not sure which option to select, first ask your [Faculty Data Steward](#) for advice. You can also check with the [privacy website](#). If you would like to contact the privacy team: privacy-tud@tudelft.nl, please bring your DMP.

- No

8B. Will you work with any other types of confidential or classified data or code as listed below? (tick all that apply)

If you are not sure which option to select, ask your [Faculty Data Steward](#) for advice.

- No, I will not work with any confidential or classified data/code

9. How will ownership of the data and intellectual property rights to the data be managed?

For projects involving commercially-sensitive research or research involving third parties, seek advice of your [Faculty Contract Manager](#) when answering this question. If this is not the case, you can use the example below.

The ownership of the datasets and intellectual property rights underlying this research will be transferred to Joost Broekens,

associate professor at Leiden University and at TU Delft. He is also a principal investigator of this study.

10. Which personal data will you process? Tick all that apply

As a manager of a survey on Amazon Mechanical Turk, the only data I can see from the participants is their Worker ID which consists of a unique number that cannot be traced back to them (so, it cannot be used for re-identification). I do not have access to their names, emails or any other personal data. IP addresses are not stored by either MTurk or Qualtrics.

Amazon MTurk collects their workers' email address and bank account information. However, this personal data is managed by Amazon Web Services, and I will not have access to it at any point.

To approve payment of the participants, I will be able to see which participant (identified by their Worker ID) gave consent to participate in the study and which did not. Upon successful completion of the survey, the participants will get a unique code generated by Qualtrics, which they will submit on MTurk. After I review their submission, and if the code they submitted is valid, I will approve payment. The payment is then processed by Amazon Web Services.

11. Please list the categories of data subjects

Adults

12. Will you be sharing personal data with individuals/organisations outside of the EEA (European Economic Area)?

- No

15. What is the legal ground for personal data processing?

- Informed consent

16. Please describe the informed consent procedure you will follow:

All study participants will be asked for their consent for taking part in the study and for data processing before the start of the survey. We will not collect the participant's written signatures since it is an online survey. The informed consent form is digital.

17. Where will you store the signed consent forms?

- Same storage solutions as explained in question 6

The answers of the digital consent forms is part of the data from the survey. Therefore, they will also be stored in the same .csv file.

18. Does the processing of the personal data result in a high risk to the data subjects?

If the processing of the personal data results in a high risk to the data subjects, it is required to perform [Data Protection Impact Assessment \(DPIA\)](#). In order to determine if there is a high risk for the data subjects, please check if any of the options below that are applicable to the processing of the personal data during your research (check all that apply).

If two or more of the options listed below apply, you will have to [complete the DPIA](#). Please get in touch with the privacy team: privacy-tud@tudelft.nl to receive support with DPIA.

If only one of the options listed below applies, your project might need a DPIA. Please get in touch with the privacy team: privacy-tud@tudelft.nl to get advice as to whether DPIA is necessary.

If you have any additional comments, please add them in the box below.

- None of the above applies

19. Did the privacy team advise you to perform a DPIA?

- No

22. What will happen with personal research data after the end of the research project?

- Other - please explain below

All data that will be collected is anonymous by default, since no personal research data will be collected.

23. How long will (pseudonymised) personal data be stored for?

- Other - please state the duration and explain the rationale below

The answers to the digital consent forms will be stored for at least two years.

24. What is the purpose of sharing personal data?

- Other - please explain below

No personal data will be shared.

25. Will your study participants be asked for their consent for data sharing?

- Yes, in consent form - please explain below what you will do with data from participants who did not consent to data sharing

If a participant does not consent, they are automatically redirected to the end of the survey and therefore cannot participate in the experiment.

V. Data sharing and long-term preservation**26. What data will be publicly shared?**

- All data (and code) produced in the project

All data and will be made available to any researcher who requests it to the principal investigators.

27. Apart from personal data mentioned in question 22, will any other data be publicly shared?

- All other non-personal data (and code) produced in the project

28. How will you share your research data (and code)?

- All data will be uploaded to 4TU.ResearchData

29. How will you share research data (and code), including the one mentioned in question 22?

- All anonymised or aggregated data, and/or all other non-personal data will be uploaded to 4TU.ResearchData with public access

30. How much of your data will be shared in a research data repository?

- 100 GB - 1 TB

31. When will the data (or code) be shared?

- At the end of the research project

32. Under what licence will be the data/code released?

- CC BY-NC-SA

VI. Data management responsibilities and resources**33. Is TU Delft the lead institution for this project?**

- Yes, leading the collaboration - please provide details of the type of collaboration and the involved parties below

TU Delft is leading the collaboration between all parties involved, which are Leiden University and TU Delft itself.

34. If you leave TU Delft (or are unavailable), who is going to be responsible for the data resulting from this project?

The principal investigators Jens Kober (J.Kober@tudelft.nl) and Joost Broekens (joost.Broekens@gmail.com, D.J.Broekens@liacs.leidenuniv.nl and D.J.Broekens@tudelft.nl).

35. What resources (for example financial and time) will be dedicated to data management and ensuring that data will be FAIR (Findable, Accessible, Interoperable, Re-usable)?

4TU.ResearchData is able to archive 1TB of data per researcher per year free of charge for all TU Delft researchers. We do not expect to exceed this and therefore there are no additional costs of long term preservation.

Planned Research Outputs

Dataset - "Emotional rating"

Dataset containing the ratings of the research videos in terms of their categorical, dimensional and appraisal-based affective qualities.

Audiovisual - "Research videos"

Videos of the appearance-constrained robot executing emotional behaviors based on the modalities of motion, light and sound.

Planned research output details

Title	Type	Anticipated release date	Initial access level	Intended repository(ies)	Anticipated file size	License	Metadata standard(s)	May contain sensitive data?	May contain PII?
Emotional rating	Dataset	2024-09-01	Restricted	None specified	100 MB	Creative Commons Attribution Share Alike 4.0 International	None specified	No	No
Research videos	Audiovisual	2024-09-01	Restricted	None specified	500 GB	Creative Commons Attribution Share Alike 4.0 International	None specified	No	No



Method Details

F.1. Wander Base Behavior

These variables are used to control the dynamics of the wander behavior, determining how long and fast the robot moves forward and turns, as well as how these actions change over time and across cycles.

Table F.1: Wander Control Variables

Parameter	Description
<code>doWander</code>	This boolean checks if the wander behavior should be executed.
<code>turnToForwardRatio</code>	Ratio defining the proportion of time spent turning versus moving forward. A higher ratio means more time turning and less time moving forward. This variable is not an input parameter but depends on the roundness and cycle rate to ensure round movements look round.
<code>turnDuration</code>	Specifies the duration for which the robot will be in the turning segment of its wander cycle. Set by <code>SetWanderDurations</code> based on <code>wanderCycleRate</code> and <code>turnToForwardRatio</code> , adjusted by <code>cycleStandardDeviation</code> to introduce variability.
<code>forwardDuration</code>	Determines how long the robot will move forward in each wander cycle. Set in <code>SetWanderDurations</code> and is a function of <code>wanderCycleRate</code> , <code>turnToForwardRatio</code> , and <code>cycleStandardDeviation</code> .
<code>lineTurnDuration</code>	Duration of the turn when the robot detects a boundary (black line). Set in <code>SetWanderDurations</code> based on the robot's speed (<code>wanderSpeed</code>) to ensure the robot has sufficient time to turn away from the boundary.
<code>targetForwardSpeed</code>	Intended speed for the robot during the forward movement segment of the wander cycle. Set in <code>SetTargetSpeeds</code> , depends on <code>wanderSpeed</code> and <code>speedStandardDeviation</code> to add randomness.
<code>targetTurnSpeed</code>	Speed at which the robot turns during the turning segment of the wander cycle. Set by <code>SetTargetSpeeds</code> , depends on <code>targetForwardSpeed</code> and <code>wanderRoundness</code> , ensuring that the speed of each wheel corresponds to the associated input roundness.
<code>acceleration</code>	Used to gradually increase or decrease the speed during the forward segment when <code>wanderSlope</code> is non-zero. Incremented or decremented during the forward movement segment to create a gradual change in speed.
<code>wanderCycle</code>	Tracks the number of complete forward-turn cycles the robot has performed during its wander behavior. Incremented after each complete cycle and influences the direction of the turn (right or left) and potentially other aspects of the next cycle.

Based on the flow diagram of the wander base behavior presented in Figure 3.2, we present a step-by-step description of the execution of wander base behavior below:

1. **Validating Wander Input:** The wander behavior executes only if `doWander` is true, indicating valid input parameters for wandering. This is determined in the `CheckValidWanderInput` method which takes all input parameters and sets the variable `doWander` to either true or false, based on whether the input parameters fall within the valid range.
2. **Control Variables Initialization:** If `doWander` is true, the wander control variables described above are initialized based on the input parameters. This is done in the `SetTargetSpeeds` and `SetWanderDurations` methods.
3. **Phase-Based Start:** The behavior starts after a delay (`wanderPhase`). This is managed by comparing the current time minus `wanderPhase` against zero.
4. **Boundary Detection Response:** If `stayInBounds` is true, the robot uses line-following sensors to stay within a defined area. To do this, the robot reads sensor values (`linefollower_2.readSensors()`) and makes decisions to turn left or right based on these readings, preventing it from crossing the edges of an area.
5. **Forward Movement:** The robot moves forward based on the value of `wanderSlope`:
 - Zero Slope: Moves forward at a constant `targetForwardSpeed` for `forwardDuration`.
 - Positive Slope (Rising): The speed increases gradually. The `acceleration` variable is incremented up to a maximum value, modifying the speed.
 - Negative Slope (Falling): The speed decreases gradually, similar to the rising slope but reducing speed over time.
6. **Turning Movement:** After the forward segment, the robot turns. The direction (right or left) is determined by the `wanderCycle` count (even number for right, odd for left). The turning speed is influenced by `targetForwardSpeed` and `targetTurnSpeed`.
7. **Cycle Control and Timing:** The `wanderCycle` variable tracks the number of completed forward-turn cycles. After completing a forward and turn sequence, `wanderCycle` is incremented, signifying the start of a new cycle.
8. **Dynamic Speed and Duration Adjustment:** At the end of each cycle, `SetTargetSpeeds` and `SetWanderDurations` are called to potentially update the speeds and durations for the next cycle, adding variability to the behavior.

F.2. Blink Base Behavior

The following control variables are initialized and updated within the code to create dynamic and customizable LED blinking patterns for the blink base behavior. The specific values of these variables are influenced by the input parameters provided to the blink behavior and may change over time as the behavior progresses through its main loop:

Table F.2: Blink Control Variables

Parameter	Description
<code>lightsOnDuration</code>	Duration for which the LED lights remain turned on during each blink cycle. Set in <code>SetBlinkDurations</code> based on <code>blinkLightsOnToOffRatio</code> , <code>blinkCycleRate</code> , and <code>blinkCycleStandardDeviation</code> .
<code>lightsOffDuration</code>	Duration for which the LED lights remain turned off during each blink cycle. Complements <code>lightsOnDuration</code> to complete the cycle. Set in <code>SetBlinkDurations</code> method.
<code>targetRedIntensity</code> , <code>targetGreenIntensity</code> , <code>targetBlueIntensity</code>	Target intensities for the red, green, and blue components of the LED during the lights-on segment. Set based on <code>blinkTemperature</code> and <code>blinkTemperatureStandardDeviation</code> in <code>SetTargetIntensities</code> method.

Continued on the next page

Table F.2: *Continued from previous page*

Parameter	Description
brightness	Controls the brightness level of the LED during the lights-on segment. Depends on <code>blinkSlope</code> ; brightness gradually changes if <code>blinkSlope</code> is non-zero, otherwise remains constant.
blinkCycle	Counter that tracks the number of completed blink cycles. Used to control adjustments such as changing target intensities or cycle durations. Incremented after each complete cycle.

Based on the flow diagram of the blink base behavior presented in Figure 3.3, we present a step-by-step description of the execution of blink base behavior below:

1. **Validating Blink Input:** Just as in the wander behavior, the blink behavior executes only if `doBlink` is true. This is determined in the `CheckValidBlinkInput` method which takes all input parameters and sets the variable `doBlink` to either true or false, based on whether the input parameters fall within the valid range.
2. **Control Variables Initialization:** If the input parameters are valid, the control variables are initialized.
3. **Phase-Based Start:** Within the main loop, the behavior takes into account the specified phase parameter. It ensures that the execution of the blink behavior starts after the given phase duration has passed. This allows for synchronization with other processes or behaviors.
4. **Alternating Lights:** The core of the blink base behavior is the alternation between lights being on and lights being off. This alternation is controlled by the parameters `blinkLightsOnToOffRatio` and `blinkCycleRate`.
5. **Brightness Control:** Depending on the `blinkSlope` parameter, which can be positive, negative, or zero, the behavior controls the brightness of the lights during the lights-on segments. This creates varying light intensities over time. If `blinkSlope` is:
 - Zero: The brightness remains constant during the lights-on segment.
 - Positive: The brightness gradually increases during the lights-on segment.
 - Negative: The brightness gradually decreases during the lights-on segment.
6. **Cycle Control and Timing:** The behavior keeps track of the blink cycles, counting how many cycles have occurred. For each cycle, it may adjust the target temperature and duration. This allows for variations in color and cycle duration over time. The behavior continues to execute until the specified duration is reached. After the duration has passed, the lights are turned off, and the behavior ends.
7. **Dynamic Temperature and Duration Adjustment:** The behavior can introduce randomness in the target temperature and cycle duration based on the provided standard deviations.

F.3. Beep Base Behavior

The quasi-periodic nature of the beep behavior and added variability allow for a wide range of sounds that can result in natural, non-linguistic utterances that are reminiscent to the sounds generated by the robots mentioned in Section 2.3.3. To control the execution of this base behavior, the following control variables are used:

Table F.3: Beep Control Variables

Parameter	Description
soundDuration	Duration of the sound (active beep) in each beep cycle. Depends on <code>soundToSilenceRatio</code> , <code>cycleRate</code> , and <code>cycleStandardDeviation</code> . Set within the <code>SetBeepDurations</code> method.

Continued on the next page

Table F.3: *Continued from previous page*

Parameter	Description
<code>silenceDuration</code>	Duration of silence (no sound) in each beep cycle. Calculated by subtracting <code>soundDuration</code> from the inverse of the cycle rate. Influenced by <code>soundToSilenceRatio</code> , <code>cycleRate</code> , and <code>cycleStandardDeviation</code> . Set within the <code>SetBeepDurations</code> method.
<code>targetPitch</code>	Desired fundamental frequency (pitch) of the beep. Influenced by the initial pitch value and <code>pitchStandardDeviation</code> . Set using the <code>SetTargetPitch</code> method.
<code>currentPitch</code>	Actual pitch of the beep during execution. Depends on <code>targetPitch</code> and may change based on the slope and semitone. Reflects the pitch variation within a beep cycle.
<code>semitone</code>	Integer value ranging from 1 to 12, used to calculate the pitch of the next note within the corresponding octave. Determined by <code>beepSlope</code> and controls pitch changes. Follows the equal temperament 12-notes-to-an-octave principle.
<code>beepCycle</code>	Counter incremented each time a complete beep cycle is executed. Helps manage the progression of the beep behavior, enabling detection of completed cycles and adjustment of parameters for subsequent cycles.

Based on the flow diagram of the beep base behavior presented in Figure 3.4, we present a step-by-step description of the execution of beep base behavior below:

- Validating Beep Input:** The beep behavior executes only if `doBeep` is true. This is determined in the `CheckValidBeepInput` method which takes all input parameters and sets the variable `doBeep` to either true or false, based on whether the input parameters fall within the valid range.
- Control Variables Initialization:** If the input parameters are valid, the control variables are initialized.
- Phase-Based Start:** Within the main loop, the behavior takes into account the specified phase parameter. It ensures that the execution of the beep behavior starts after the given phase duration has passed.
- Beep Generation Based on Beep Slope:** The behavior differentiates between three cases based on the `beepSlope` parameter:
 - If `beepSlope` is 0 (constant pitch), it plays the sound at the `currentPitch` for the duration specified by `soundDuration`.
 - If `beepSlope` is positive (rising intonation), it plays the sound at the `currentPitch` for a duration corresponding to a twelfth of the `soundDuration`. After this duration, it increases the `currentPitch` to the next rising semitone within the octave using the following formula:
 - If `beepSlope` is negative (falling intonation), it plays the sound at the `currentPitch` for a duration corresponding to a twelfth of the `soundDuration`. After this duration, it decreases the `currentPitch` to the next falling semitone within the octave.
- Random Sound Generation:** The behavior checks if it should play either a random note or maintain silence based on the given `beepRandomSoundProbability`. If the probability condition is met, it plays a random sound with pitch variation based on `beepPitchStandardDeviation`.
- Cycle Control and Timing:** The behavior checks if the current beep cycle is completed based on timing. If so, it proceeds to the next cycle, increments `beepCycle` by one, and resets `semitone` to 1.
- Dynamic Pitch and Duration Adjustment:** The behavior can introduce randomness in the target pitch based on `beepPitch` by adding Gaussian variability with `beepPitchStandardDeviation`, and adjusts sound and silence durations in the `SetBeepDurations`.

F.4. Controlled Variables

In the table below, the chosen values for the controlled variables are shown:

Table F.4: Controlled Variables

Parameter	Value
duration	20
stayInBounds	True
wanderSlope	0
wanderCycleStandardDeviation	0.4
wanderSpeedStandardDeviation	0
wanderPhase	0
blinkLightsOnToOffRatio	0.85
blinkCycleStandardDeviation	0.325
blinkTemperatureStandardDeviation	0.025
blinkPhase	0
beepCycleStandardDeviation	0.2
beepSoundToSilenceRatio	0.625
beepPitchStandardDeviation	67.5
beepRandomSoundProbability	0
beepPhase	0

F.5. Sobol Sequence Sampling

The Sobol sequence, specifically, generates values for each parameter such that the distribution of points is evenly spread across the multidimensional space. Mathematically, a Sobol sequence is designed to fill the space more uniformly than unstructured random numbers. The formula for generating a new sample in a Sobol sequence is:

$$x_i = \frac{x_{i-1} \oplus a_n}{2^n} \quad (\text{F.1})$$

Where x_i is the new sample, x_{i-1} is the previous sample, \oplus denotes a bitwise exclusive or operation, a_n is a direction number, and n refers to the dimension.

This method helps ensure that each sample point is spaced to minimize correlation, thereby maintaining the independence among variables. This property is crucial when examining the impact of each independent variable without overlap or undue influence from others.

Another important consideration of Sobol sequences, is that they are a quadrature rule and thus lose their balance properties if one uses a sample size that is not a power of 2 [62]. Having this into consideration, it was discovered by inspection that for a 9-dimensional independent variable space, we would need at least $2^9 = 512$ samples to ensure the balance properties of the samples.

The steps taken to generate and scale Sobol sequences to fit the desired range of each independent variable are as follows:

- **Initialization:** Setting up the Sobol sequence generator with the specified number of variables and enabling scrambling.
- **Sample Generation:** Generating 512 samples using the Sobol sequence.
- **Scaling:** Scaling the samples to specific lower and upper bounds for each variable, ensuring that the generated values are within practical and meaningful ranges.
- **Data Creation:** Creating a data structure (DataFrame) to hold the scaled samples with appropriate column names.

- **Rounding:** Adjusting the `beepSlope` and `blinkSlope` values to be integers, as these parameters need to be represented as discrete levels.
- **Saving:** Storing the final dataset in a CSV file for further analysis.

To illustrate the effectiveness of this sampling method, in Figure F.1 we show the histograms of each independent variable to demonstrate that the variables are uniformly sampled.

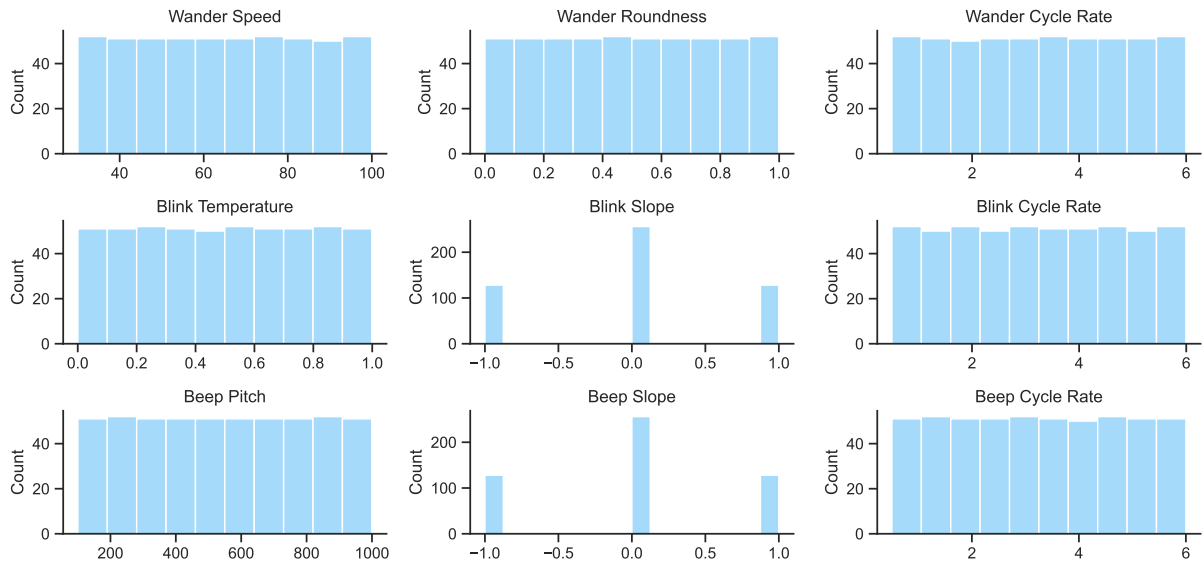


Figure F.1: Distribution of the independent variables

In Figure F.2, a heatmap of the correlation matrix of the independent variables is plotted. Only the lower half is shown as the matrix is symmetric. As is visible in the heatmap, the diagonal entries are 1, and the non-diagonal entries are very small and negligible (rounded to zero), indicating minimal correlation among the variables. This is an important observation, as it ensures that we do not introduce any multicollinearity among the independent variables which could affect the results of our analysis.

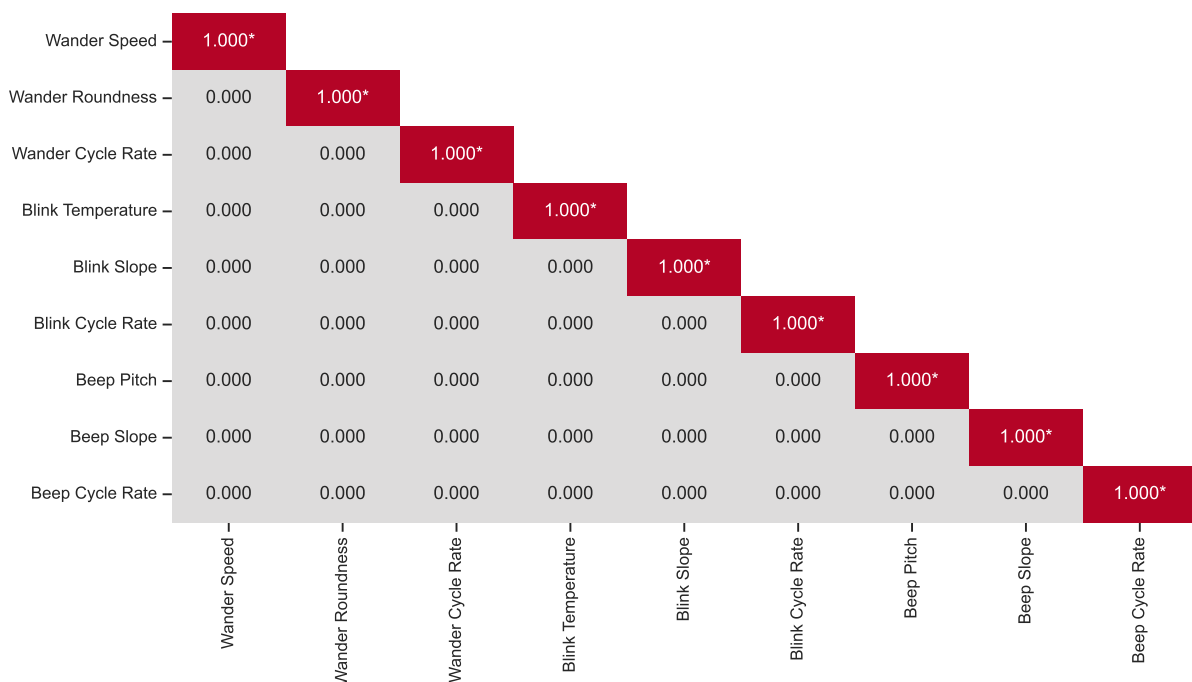


Figure F.2: Correlation matrix of the independent variables

F.6. Detailed Micro-Hypotheses

To be able to test if there exists significant correlations between the input parameters of the wander base behavior (`wanderSpeed`, `wanderRoundness` and `wanderCycleRate`) with the nine dependent variables, we define the following null hypotheses:

Table F.5: Detailed micro-hypotheses for RQ1

ID	Null Hypothesis
H1.1.1	There is no significant correlation between <code>wanderSpeed</code> and joy intensity.
H1.1.2	There is no significant correlation between <code>wanderSpeed</code> and sadness intensity.
H1.1.3	There is no significant correlation between <code>wanderSpeed</code> and fear intensity.
H1.1.4	There is no significant correlation between <code>wanderSpeed</code> and anger intensity.
H1.1.5	There is no significant correlation between <code>wanderSpeed</code> and disgust intensity.
H1.1.6	There is no significant correlation between <code>wanderSpeed</code> and surprise intensity.
H1.1.7	There is no significant correlation between <code>wanderSpeed</code> and pleasure.
H1.1.8	There is no significant correlation between <code>wanderSpeed</code> and arousal.
H1.1.9	There is no significant correlation between <code>wanderSpeed</code> and dominance.
H1.2.1	There is no significant correlation between <code>wanderRoundness</code> and joy intensity.
H1.2.2	There is no significant correlation between <code>wanderRoundness</code> and sadness intensity.
H1.2.3	There is no significant correlation between <code>wanderRoundness</code> and fear intensity.
H1.2.4	There is no significant correlation between <code>wanderRoundness</code> and anger intensity.
H1.2.5	There is no significant correlation between <code>wanderRoundness</code> and disgust intensity.
H1.2.6	There is no significant correlation between <code>wanderRoundness</code> and surprise intensity.
H1.2.7	There is no significant correlation between <code>wanderRoundness</code> and pleasure.
H1.2.8	There is no significant correlation between <code>wanderRoundness</code> and arousal.
H1.2.9	There is no significant correlation between <code>wanderRoundness</code> and dominance.
H1.3.1	There is no significant correlation between <code>wanderCycleRate</code> and joy intensity.
H1.3.2	There is no significant correlation between <code>wanderCycleRate</code> and sadness intensity.
H1.3.3	There is no significant correlation between <code>wanderCycleRate</code> and fear intensity.
H1.3.4	There is no significant correlation between <code>wanderCycleRate</code> and anger intensity.
H1.3.5	There is no significant correlation between <code>wanderCycleRate</code> and disgust intensity.
H1.3.6	There is no significant correlation between <code>wanderCycleRate</code> and surprise intensity.
H1.3.7	There is no significant correlation between <code>wanderCycleRate</code> and pleasure.
H1.3.8	There is no significant correlation between <code>wanderCycleRate</code> and arousal.
H1.3.9	There is no significant correlation between <code>wanderCycleRate</code> and dominance.

We do the same for the the input parameters of the blink base behavior (`blinkTemperature`, `blinkSlope` and `blinkCycleRate`):

Table F.6: Detailed micro-hypotheses for RQ2

ID	Null Hypothesis
H2.1.1	There is no significant correlation between <code>blinkTemperature</code> and joy intensity.
H2.1.2	There is no significant correlation between <code>blinkTemperature</code> and sadness intensity.
H2.1.3	There is no significant correlation between <code>blinkTemperature</code> and fear intensity.
H2.1.4	There is no significant correlation between <code>blinkTemperature</code> and anger intensity.
H2.1.5	There is no significant correlation between <code>blinkTemperature</code> and disgust intensity.

Continued on the next page

Table F.6: *Continued from previous page*

ID	Null Hypothesis
H2.1.6	There is no significant correlation between <code>blinkTemperature</code> and surprise intensity.
H2.1.7	There is no significant correlation between <code>blinkTemperature</code> and pleasure.
H2.1.8	There is no significant correlation between <code>blinkTemperature</code> and arousal.
H2.1.9	There is no significant correlation between <code>blinkTemperature</code> and dominance.
H2.2.1	There is no significant correlation between <code>blinkSlope</code> and joy intensity.
H2.2.2	There is no significant correlation between <code>blinkSlope</code> and sadness intensity.
H2.2.3	There is no significant correlation between <code>blinkSlope</code> and fear intensity.
H2.2.4	There is no significant correlation between <code>blinkSlope</code> and anger intensity.
H2.2.5	There is no significant correlation between <code>blinkSlope</code> and disgust intensity.
H2.2.6	There is no significant correlation between <code>blinkSlope</code> and surprise intensity.
H2.2.7	There is no significant correlation between <code>blinkSlope</code> and pleasure.
H2.2.8	There is no significant correlation between <code>blinkSlope</code> and arousal.
H2.2.9	There is no significant correlation between <code>blinkSlope</code> and dominance.
H2.3.1	There is no significant correlation between <code>blinkCycleRate</code> and joy intensity.
H2.3.2	There is no significant correlation between <code>blinkCycleRate</code> and sadness intensity.
H2.3.3	There is no significant correlation between <code>blinkCycleRate</code> and fear intensity.
H2.3.4	There is no significant correlation between <code>blinkCycleRate</code> and anger intensity.
H2.3.5	There is no significant correlation between <code>blinkCycleRate</code> and disgust intensity.
H2.3.6	There is no significant correlation between <code>blinkCycleRate</code> and surprise intensity.
H2.3.7	There is no significant correlation between <code>blinkCycleRate</code> and pleasure.
H2.3.8	There is no significant correlation between <code>blinkCycleRate</code> and arousal.
H2.3.9	There is no significant correlation between <code>blinkCycleRate</code> and dominance.

And finally, we define the null hypotheses to test if there exist significant correlations between the input parameters of the beep base behavior (`beepPitch`, `beepSlope` and `beepCycleRate`) and the dependent variables:

Table F.7: Detailed micro-hypotheses for RQ3

ID	Null Hypothesis
H3.1.1	There is no significant correlation between <code>beepPitch</code> and joy intensity.
H3.1.2	There is no significant correlation between <code>beepPitch</code> and sadness intensity.
H3.1.3	There is no significant correlation between <code>beepPitch</code> and fear intensity.
H3.1.4	There is no significant correlation between <code>beepPitch</code> and anger intensity.
H3.1.5	There is no significant correlation between <code>beepPitch</code> and disgust intensity.
H3.1.6	There is no significant correlation between <code>beepPitch</code> and surprise intensity.
H3.1.7	There is no significant correlation between <code>beepPitch</code> and pleasure.
H3.1.8	There is no significant correlation between <code>beepPitch</code> and arousal.
H3.1.9	There is no significant correlation between <code>beepPitch</code> and dominance.
H3.2.1	There is no significant correlation between <code>beepSlope</code> and joy intensity.
H3.2.2	There is no significant correlation between <code>beepSlope</code> and sadness intensity.
H3.2.3	There is no significant correlation between <code>beepSlope</code> and fear intensity.
H3.2.4	There is no significant correlation between <code>beepSlope</code> and anger intensity.
H3.2.5	There is no significant correlation between <code>beepSlope</code> and disgust intensity.

Continued on the next page

Table F.7: *Continued from previous page*

ID	Null Hypothesis
H3.2.6	There is no significant correlation between <code>beepSlope</code> and surprise intensity.
H3.2.7	There is no significant correlation between <code>beepSlope</code> and pleasure.
H3.2.8	There is no significant correlation between <code>beepSlope</code> and arousal.
H3.2.9	There is no significant correlation between <code>beepSlope</code> and dominance.
H3.3.1	There is no significant correlation between <code>beepCycleRate</code> and joy intensity.
H3.3.2	There is no significant correlation between <code>beepCycleRate</code> and sadness intensity.
H3.3.3	There is no significant correlation between <code>beepCycleRate</code> and fear intensity.
H3.3.4	There is no significant correlation between <code>beepCycleRate</code> and anger intensity.
H3.3.5	There is no significant correlation between <code>beepCycleRate</code> and disgust intensity.
H3.3.6	There is no significant correlation between <code>beepCycleRate</code> and surprise intensity.
H3.3.7	There is no significant correlation between <code>beepCycleRate</code> and pleasure.
H3.3.8	There is no significant correlation between <code>beepCycleRate</code> and arousal.
H3.3.9	There is no significant correlation between <code>beepCycleRate</code> and dominance.

To test the significance of the difference in performance between pairs of baseline and optimized models, we define the micro-hypothesis of H4:

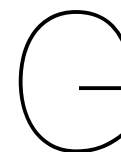
Table F.8: Detailed micro-hypotheses of H4

ID	Null Hypothesis
H4.1.1	The linear regression models of the six basic emotions trained with X do not achieve significantly lower MSE and higher R^2 than the mean predictor of the six basic emotions.
H4.1.2	The linear regression models of the PAD dimensions trained with X do not achieve significantly lower MSE and higher R^2 than the mean predictor of the PAD dimensions.
H4.2.1	The linear regression models of the six basic emotions trained with X_{int} do not achieve significantly lower MSE and higher R^2 than the linear regression models of the six basic emotions trained with X .
H4.2.2	The linear regression models of the PAD dimensions trained with X_{int} do not achieve significantly lower MSE and higher R^2 than the linear regression models of the PAD dimensions trained with X .
H4.3.1	The linear regression models of the six basic emotions trained with X_{gmm} do not achieve significantly lower MSE and higher R^2 than the linear regression models of the six basic emotions trained with X .
H4.3.2	The linear regression models of the PAD dimensions trained with X_{gmm} do not achieve significantly lower MSE and higher R^2 than the linear regression models of the PAD dimensions trained with X .
H4.4.1	The linear regression models of the six basic emotions trained with X_{gmm_int} do not achieve significantly lower MSE and higher R^2 than the linear regression models of the six basic emotions trained with X_{gmm} .
H4.4.2	The linear regression models of the PAD dimensions trained with X_{gmm_int} do not achieve significantly lower MSE and higher R^2 than the linear regression models of the PAD dimensions trained with X_{gmm} .
H4.5.1	The random forest regression models of the six basic emotions trained with X do not achieve significantly lower MSE and higher R^2 than the linear regression models of the six basic emotions trained with X

Continued on the next page

Table F.8: *Continued from previous page*

ID	Null Hypothesis
H4.5.2	The random forest regression models of the PAD dimensions trained with X do not achieve significantly lower MSE and higher R^2 than the linear regression models of the PAD dimensions trained with X
H4.6.1	The random forest regression models of the six basic emotions trained with X_{int} do not achieve significantly lower MSE and higher R^2 than the random forest regression models of the six basic emotions trained with X .
H4.6.2	The random forest regression models of the PAD dimensions trained with X_{int} do not achieve significantly lower MSE and higher R^2 than the random forest regression models of the PAD dimensions trained with X .
H4.7.1	The random forest regression models of the six basic emotions trained with X_{gmm} do not achieve significantly lower MSE and higher R^2 than the random forest regression models of the six basic emotions trained with X .
H4.7.2	The random forest regression models of the PAD dimensions trained with X_{gmm} do not achieve significantly lower MSE and higher R^2 than the random forest regression models of the PAD dimensions trained with X .
H4.8.1	The random forest regression models of the six basic emotions trained with X_{gmm_int} do not achieve significantly lower MSE and higher R^2 than the random forest regression models of the six basic emotions trained with X_{gmm} .
H4.8.2	The random forest regression models of the PAD dimensions trained with X_{gmm_int} do not achieve significantly lower MSE and higher R^2 than the random forest regression models of the PAD dimensions trained with X_{gmm} .



Results Details

G.1. Exploratory Data Analysis

To illustrate the effect of the aggregation in terms of noise reduction of the dependent variables, Table G.1 presents descriptive statistics of the 9 dependent variables pre-aggregation. This can be compared to the descriptive statistics of the 9 dependent variables post-aggregation in Table 4.1.

Table G.1: Descriptive statistics of the dependent variables pre-aggregation

	Mean	Std. Dev.	Min.	Q1	Median	Q3	Max.
Joy Intensity	1.727	1.624	0.000	0.000	1.000	3.000	5.000
Sadness Intensity	1.439	1.479	0.000	0.000	1.000	3.000	5.000
Anger Intensity	0.828	1.153	0.000	0.000	0.000	1.000	5.000
Fear Intensity	1.305	1.502	0.000	0.000	1.000	2.000	5.000
Disgust Intensity	0.459	0.670	0.000	0.000	0.000	1.000	2.000
Surprise Intensity	1.416	1.473	0.000	0.000	1.000	3.000	5.000
Pleasure	4.459	2.314	1.000	3.000	4.000	6.000	9.000
Arousal	5.139	2.063	1.000	4.000	5.000	7.000	9.000
Dominance	4.355	2.105	1.000	3.000	4.000	6.000	9.000

To understand the distribution of our data, we present the histograms of the aggregated dependent variables in Figure G.1 and Figure G.2.

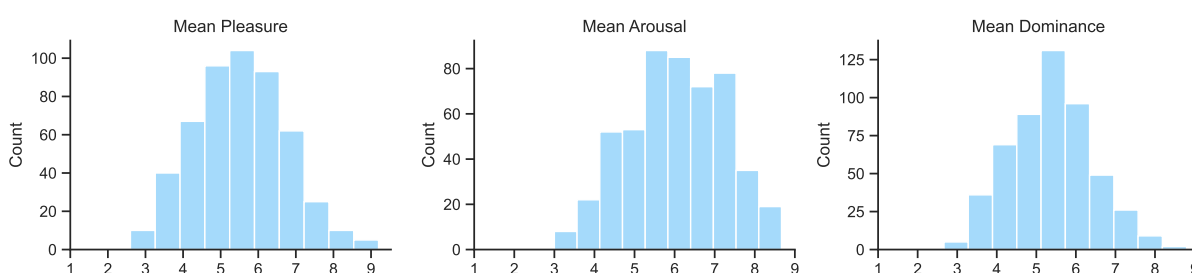


Figure G.1: Distribution of the PAD dimensions variables based on the aggregated dataset

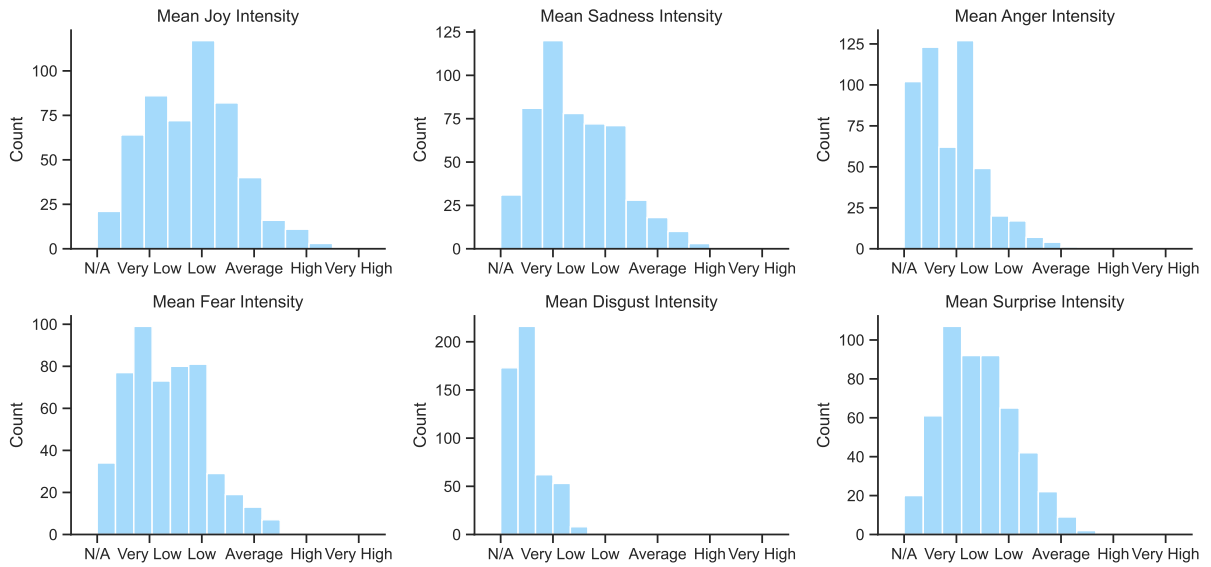


Figure G.2: Distribution of the emotional intensity variables based on the aggregated dataset

To further illustrate the spread of the aggregated dependent variables, we present the box plots for each variable in Figures G.3 and G.4.

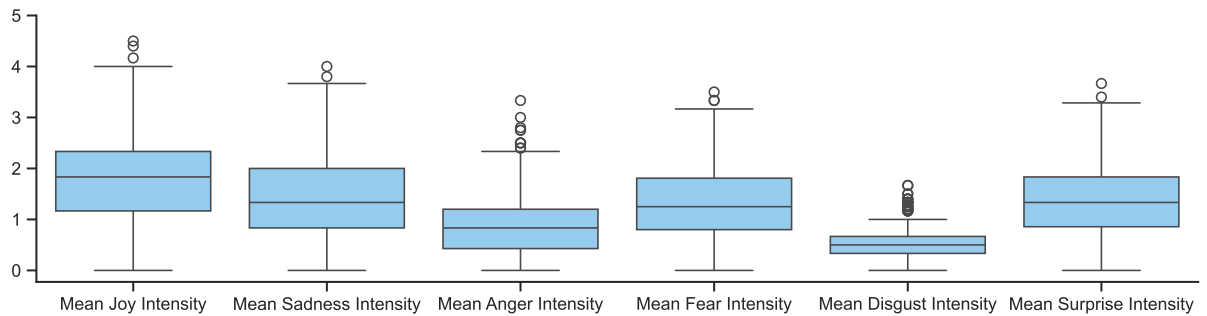


Figure G.3: Box plots of the emotional intensity variables based on the aggregated dataset

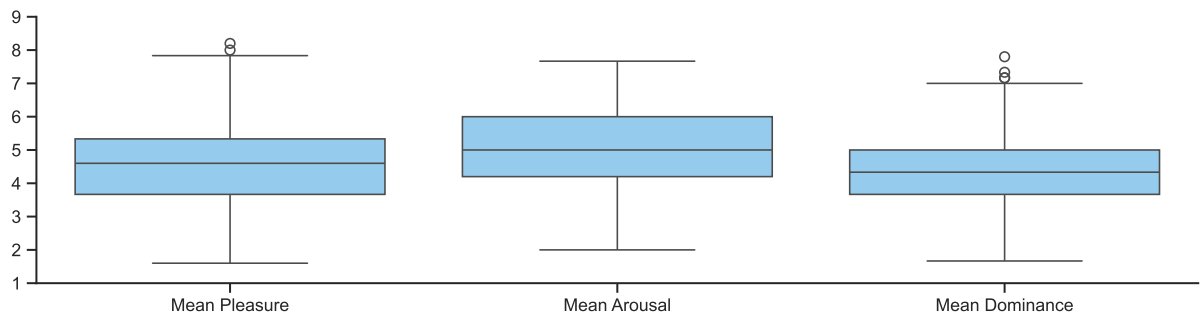


Figure G.4: Box plots of the PAD dimensions variables based on the aggregated dataset

G.2. Correlation Analysis

To understand the relationships among the dependent variables themselves, we provide the self-correlation matrices for the dependent variables. The self-correlation matrix of the aggregated dependent variables is shown in Figure G.5:

To illustrate the strength of the monotonic relationships between the independent and the dependent variables, the correlation matrix is displayed in a heatmap in Figure G.6. The significant correlations

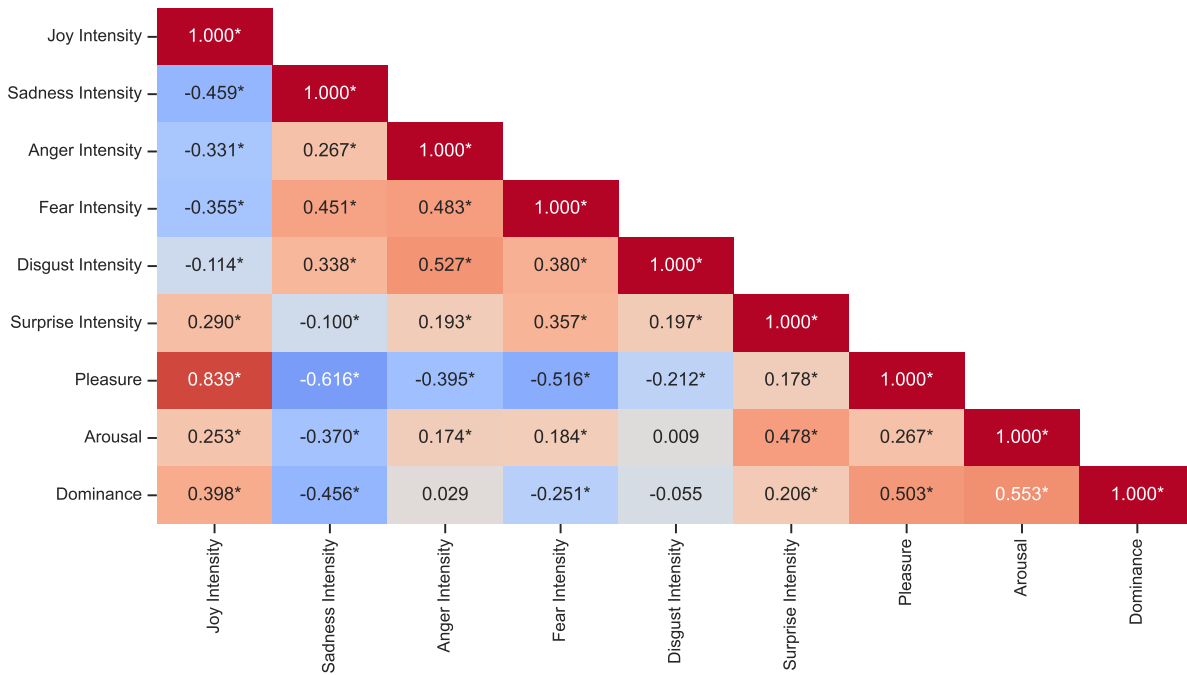


Figure G.5: Self-correlation matrix of the dependent variables based on the aggregated dataset

according to Spearman’s rank-order correlation coefficient test (without any multiple testing correction) are annotated with an asterisk.

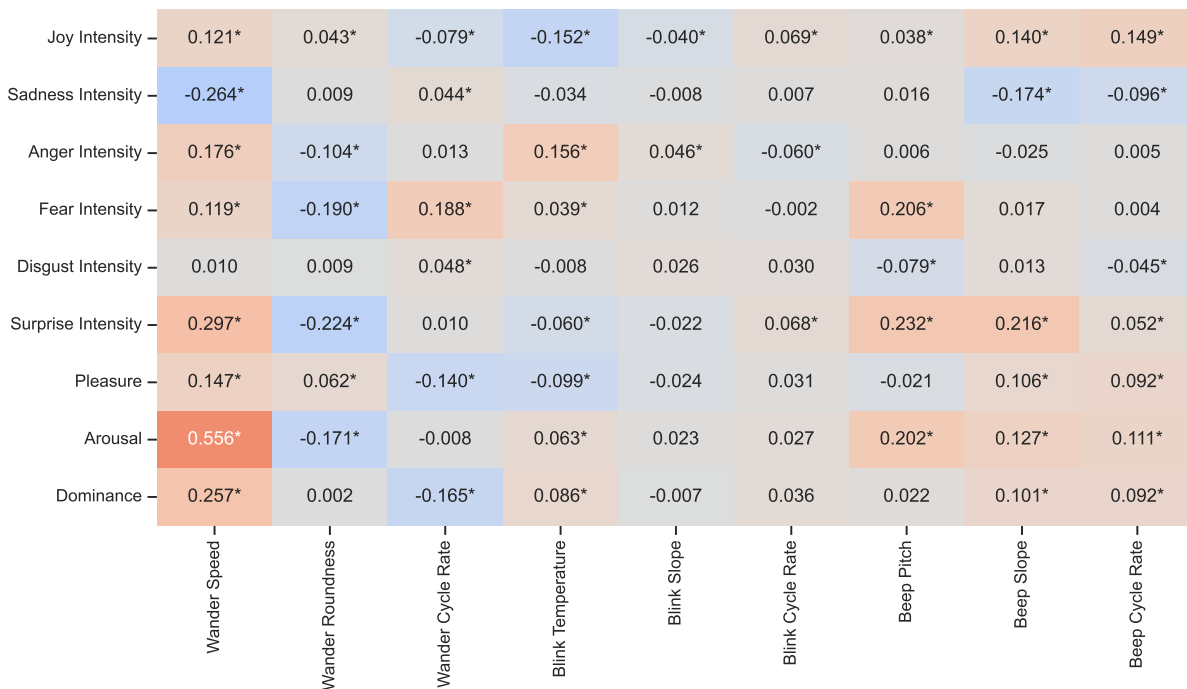


Figure G.6: Correlation matrix of the independent and the aggregated dependent variables

G.3. Regression Analysis Performance Plots

In this section we will include the scatter plots for all 36 regression models. To keep this section structured, we will divide it by subsections, each corresponding to a target dependent variable.

G.3.1. Joy Intensity Performance Scatter Plots

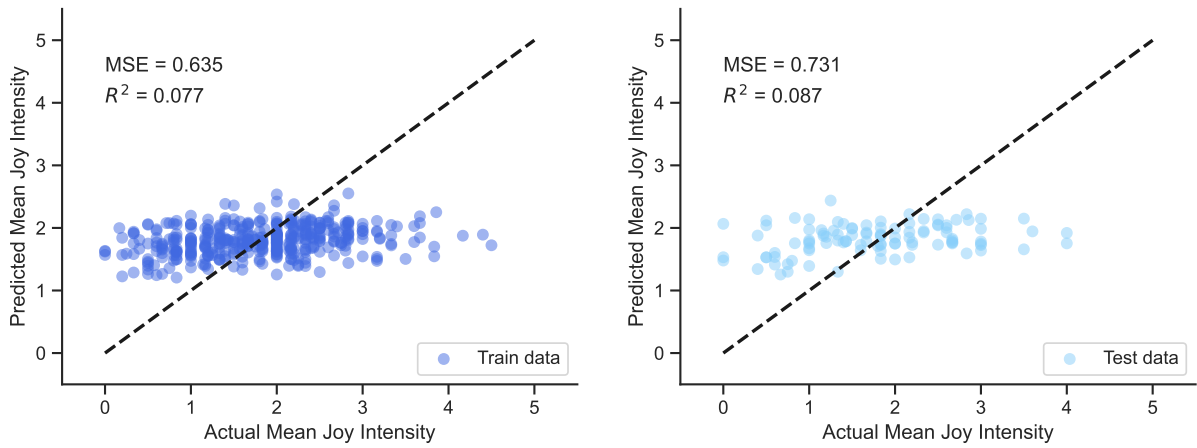


Figure G.7: Scatterplots of the linear model trained with x for joy intensity

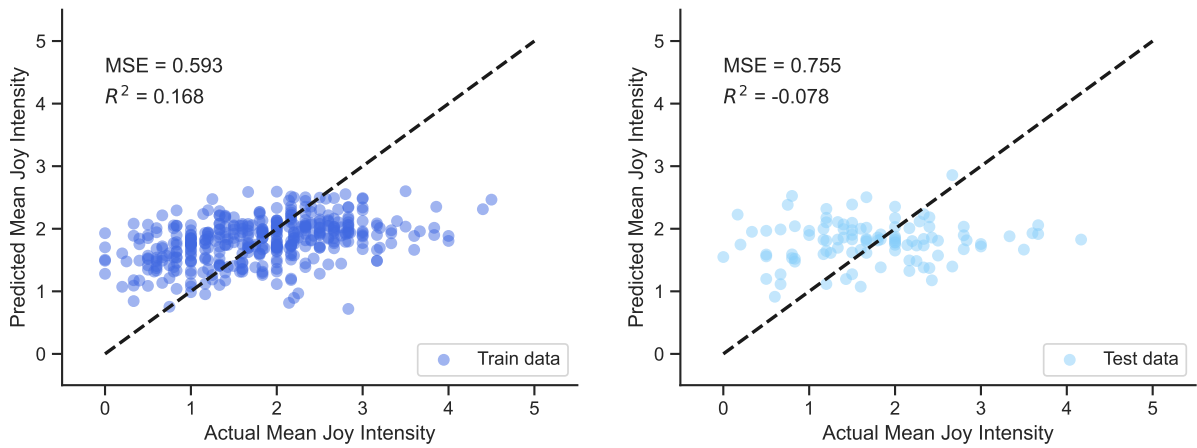


Figure G.8: Scatterplots of the linear model trained with X_{int} for joy intensity

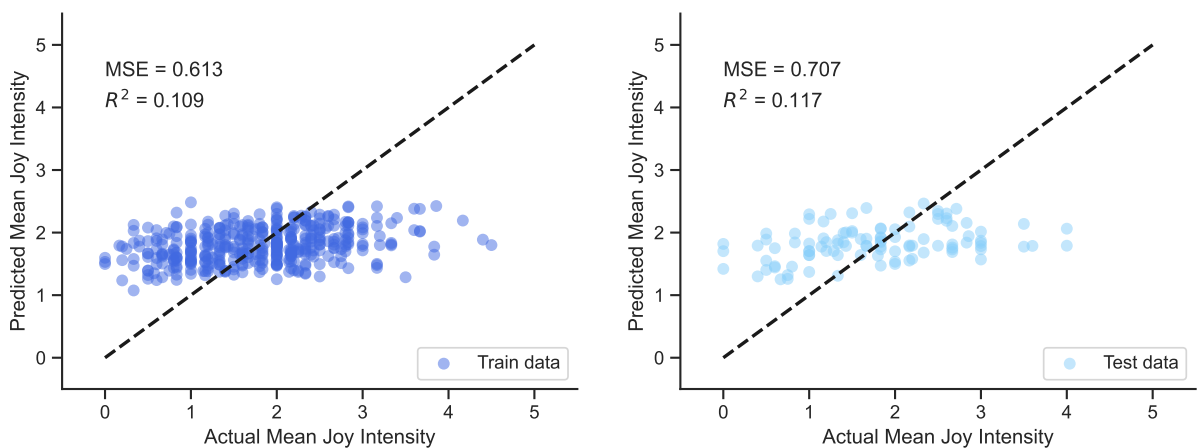


Figure G.9: Scatterplots of the linear model trained with X_{gmm} for joy intensity

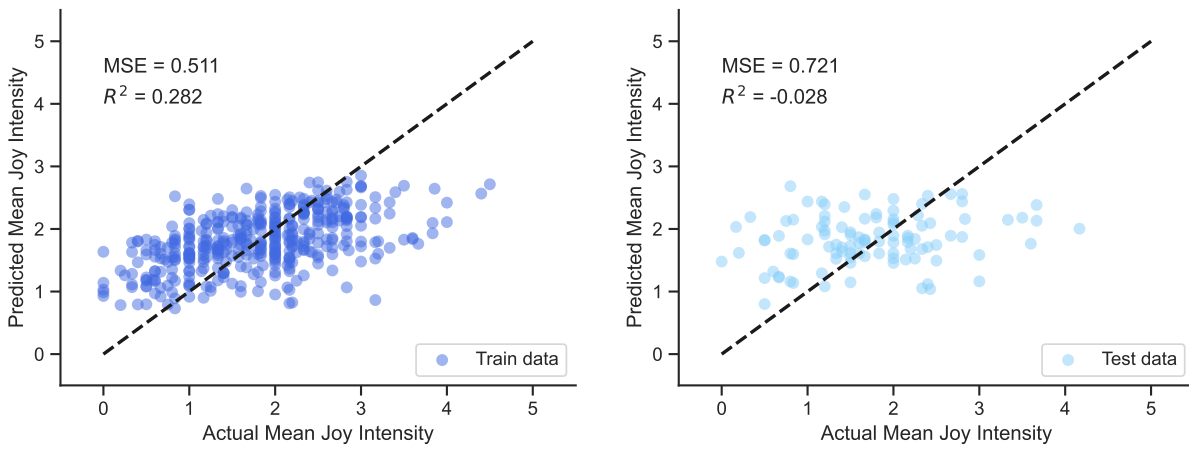


Figure G.10: Scatterplots of the linear model trained with X_{gmm_int} for joy intensity

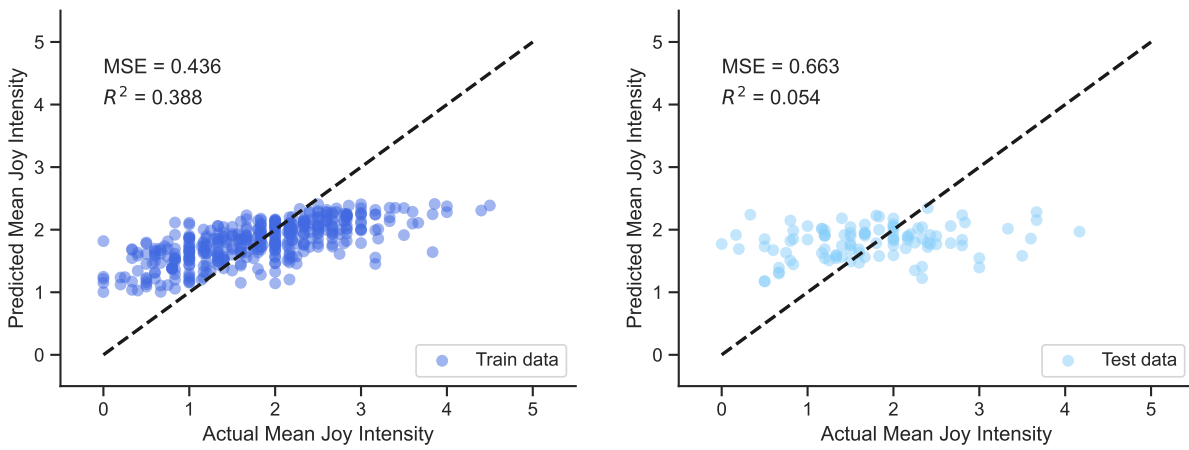


Figure G.11: Scatterplots of the random forest model trained with X for joy intensity

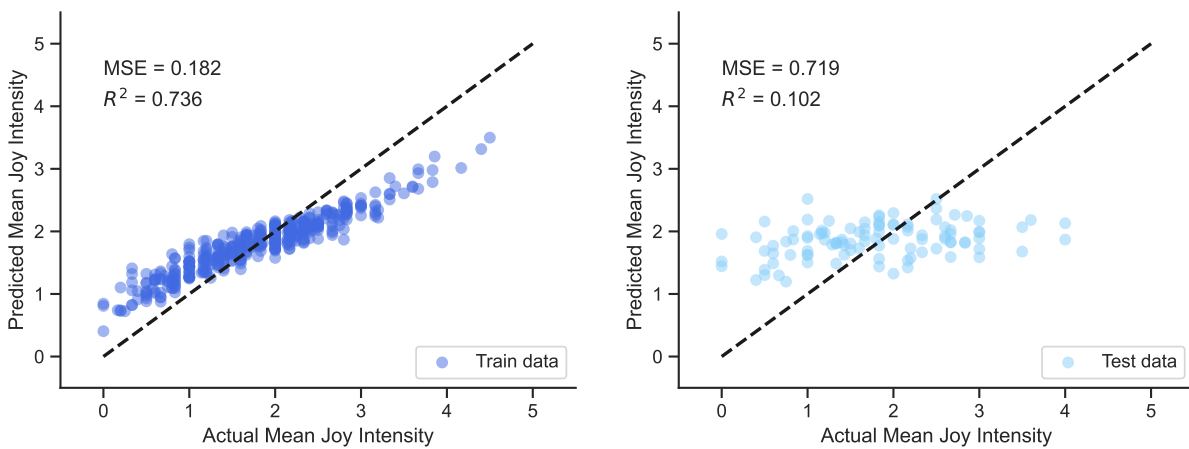


Figure G.12: Scatterplots of the random forest model trained with X_{int} for joy intensity

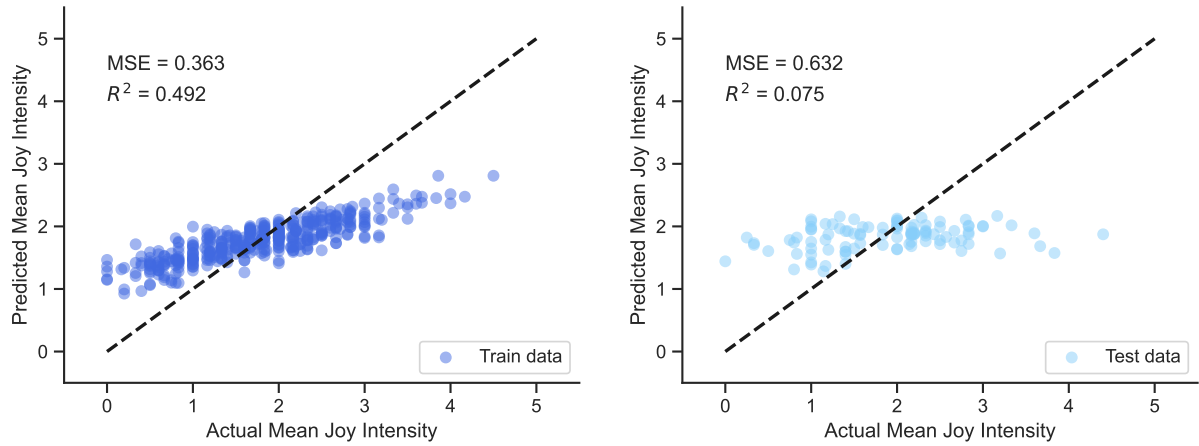


Figure G.13: Scatterplots of the random forest model trained with X_{gmm} for joy intensity

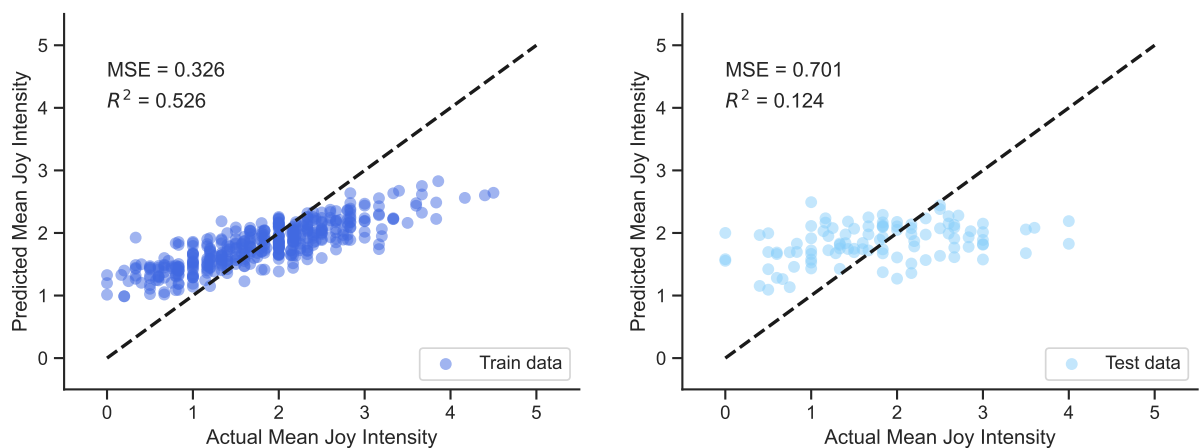


Figure G.14: Scatterplots of the random forest model trained with X_{gmm_int} for joy intensity

G.3.2. Sadness Intensity Performance Scatter Plots

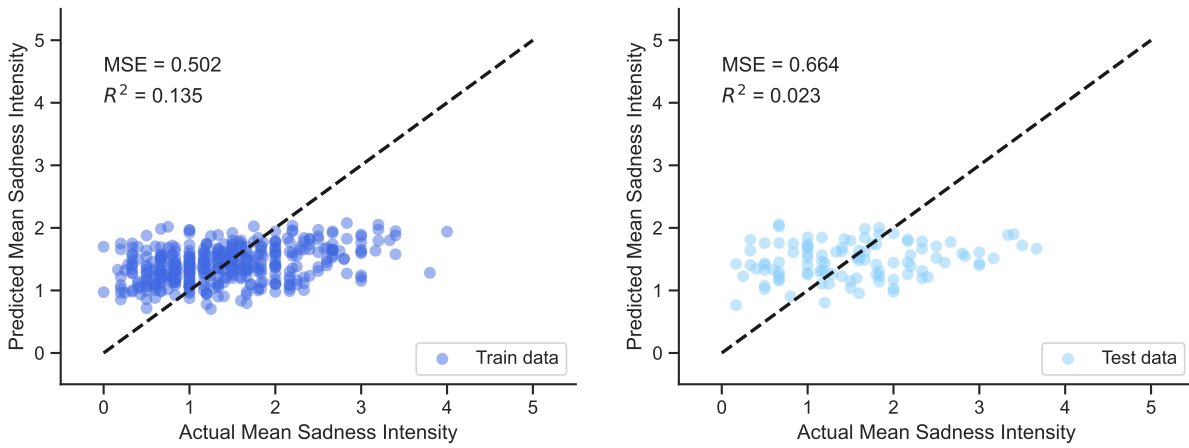


Figure G.15: Scatterplots of the linear model trained with x for sadness intensity

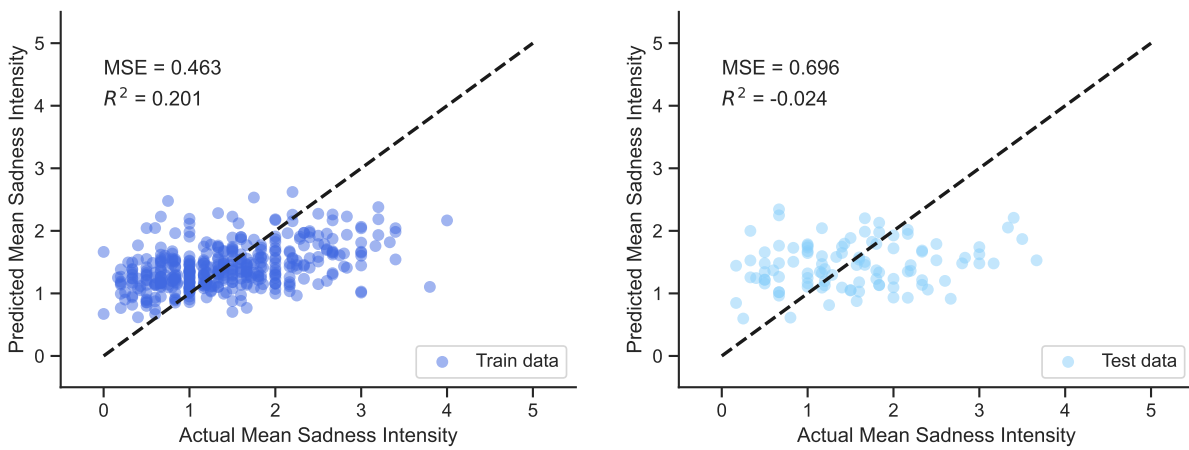


Figure G.16: Scatterplots of the linear model trained with X_{int} for sadness intensity

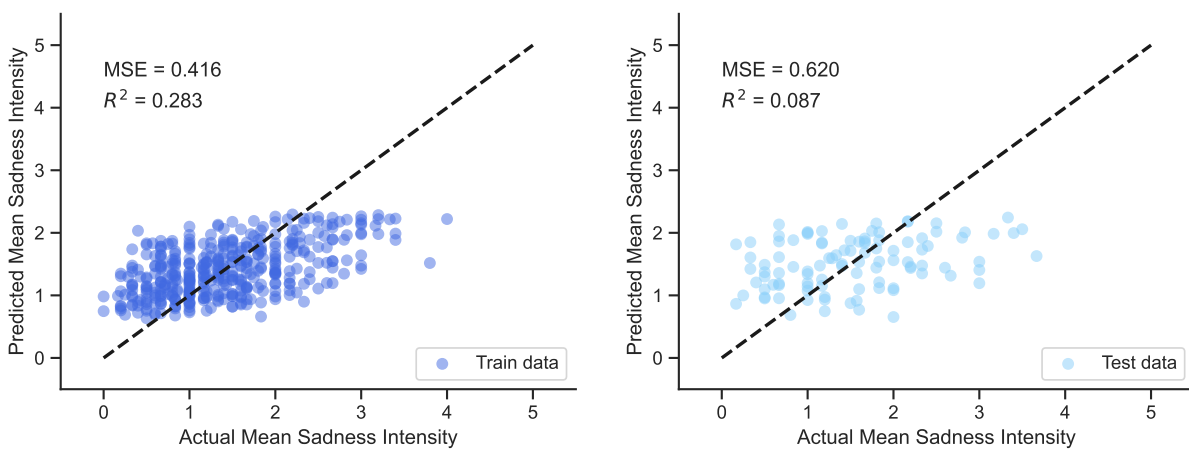


Figure G.17: Scatterplots of the linear model trained with X_{gmm} for sadness intensity

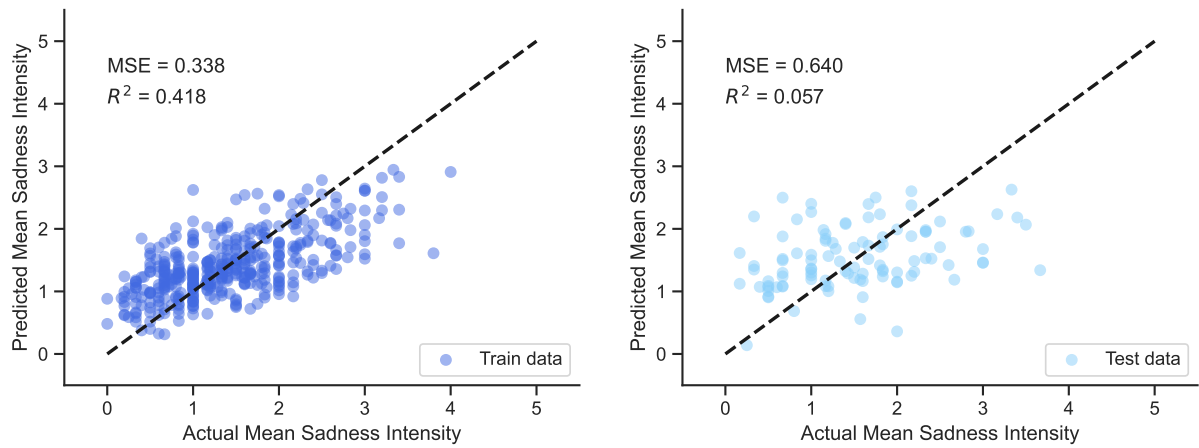


Figure G.18: Scatterplots of the linear model trained with X_{gmm_int} for sadness intensity

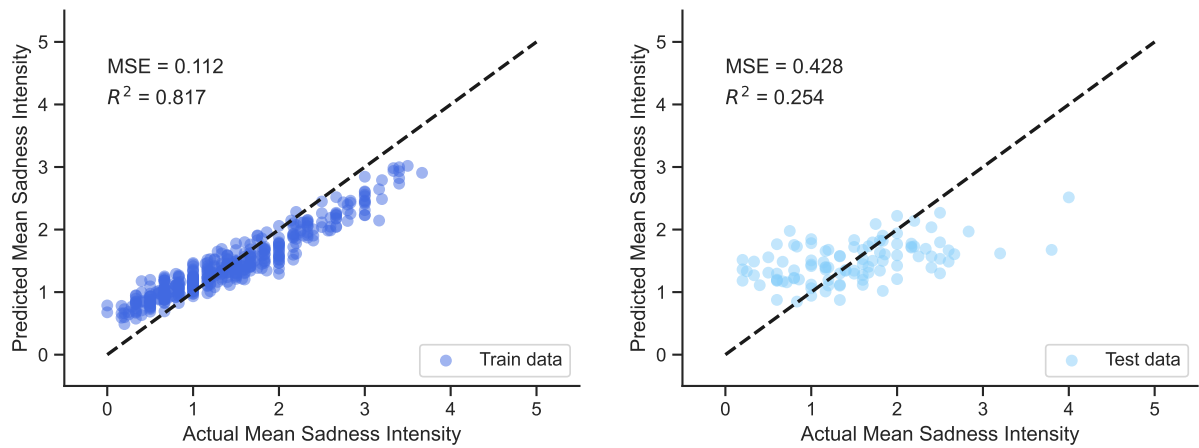


Figure G.19: Scatterplots of the random forest model trained with X for sadness intensity

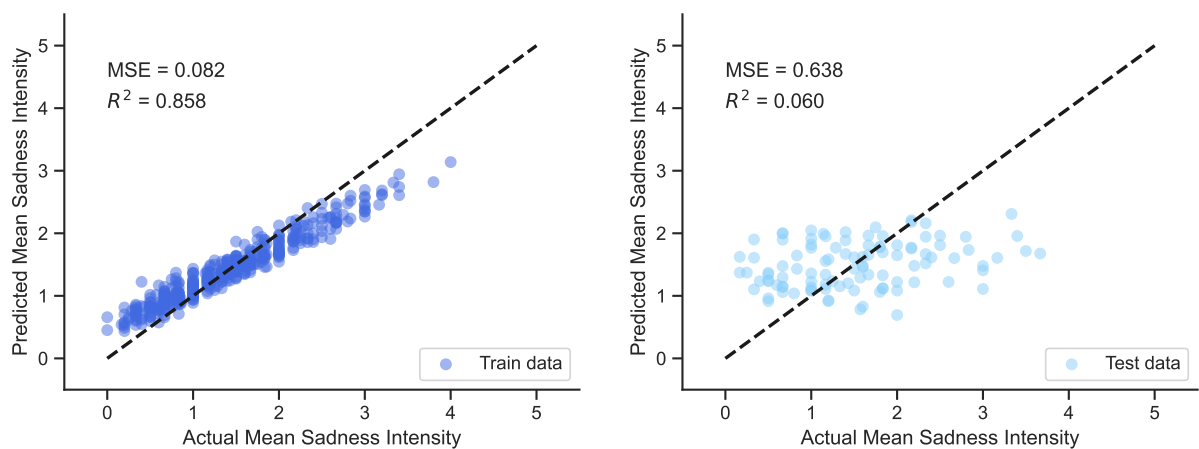


Figure G.20: Scatterplots of the random forest model trained with X_{int} for sadness intensity

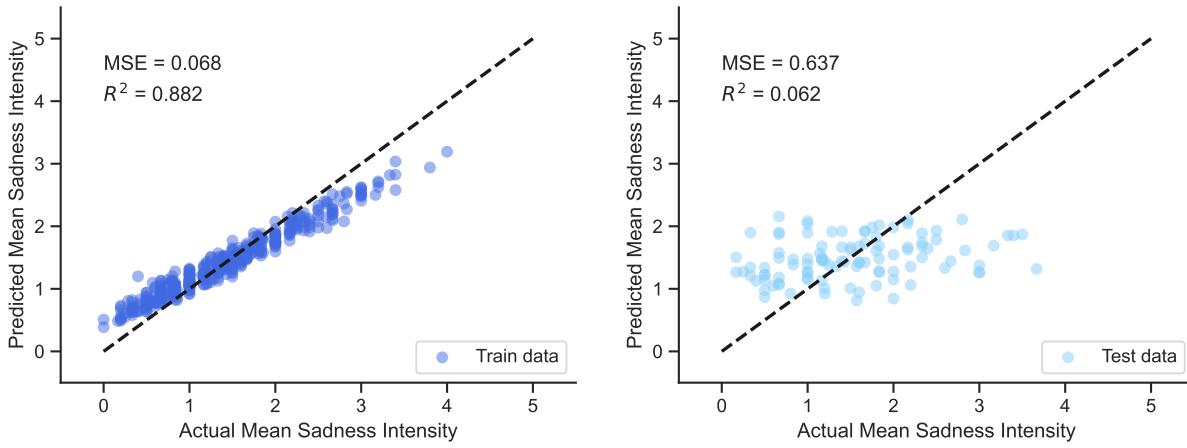


Figure G.21: Scatterplots of the random forest model trained with X_{gmm} for sadness intensity

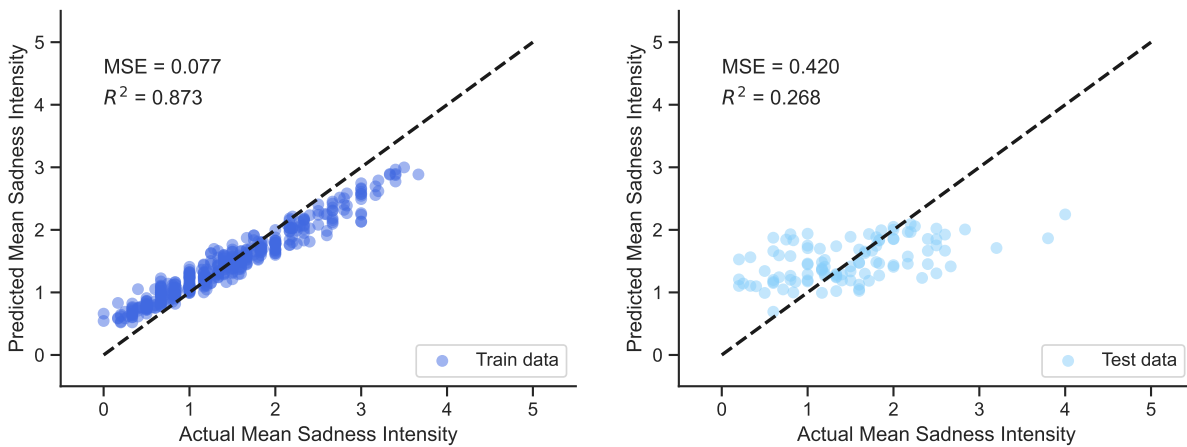


Figure G.22: Scatterplots of the random forest model trained with X_{gmm_int} for sadness intensity

G.3.3. Fear Intensity Performance Scatter Plots

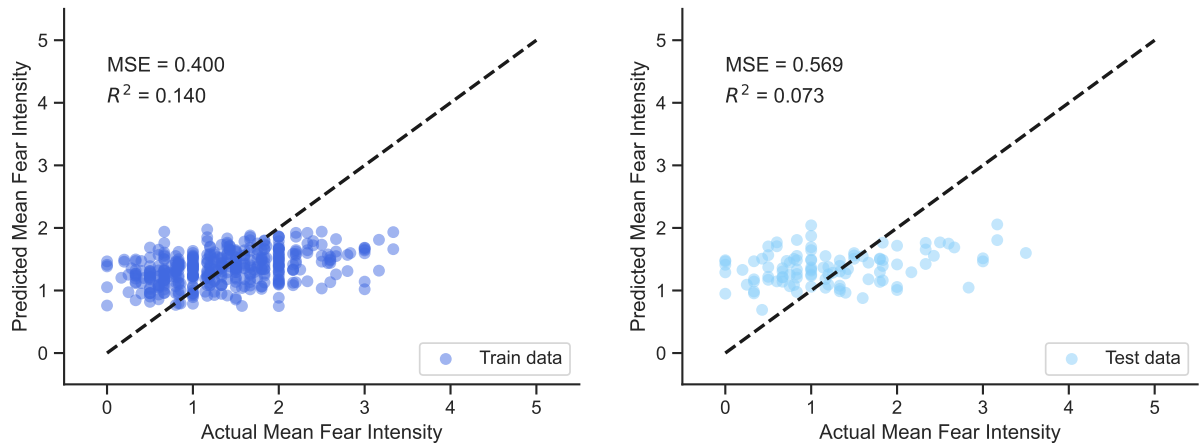


Figure G.23: Scatterplots of the linear model trained with x for fear intensity

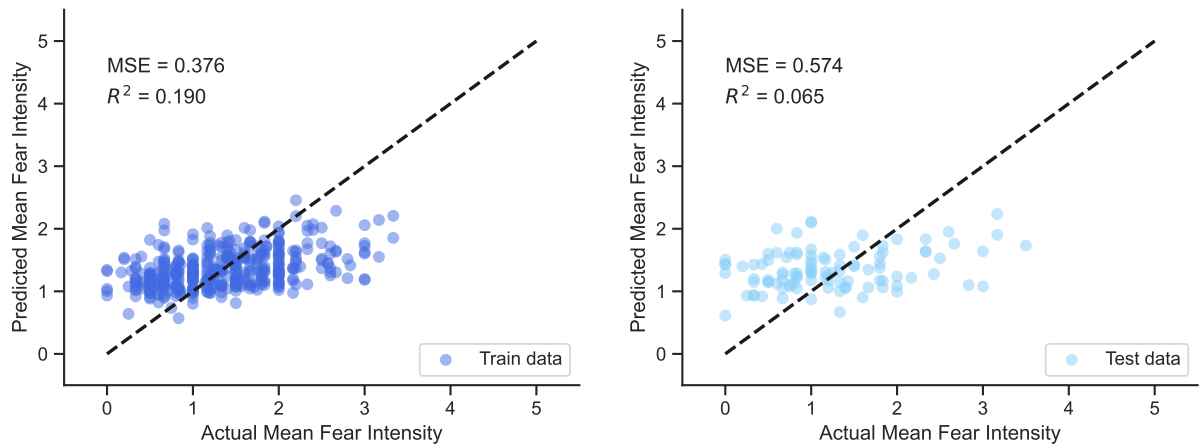


Figure G.24: Scatterplots of the linear model trained with X_{int} for fear intensity

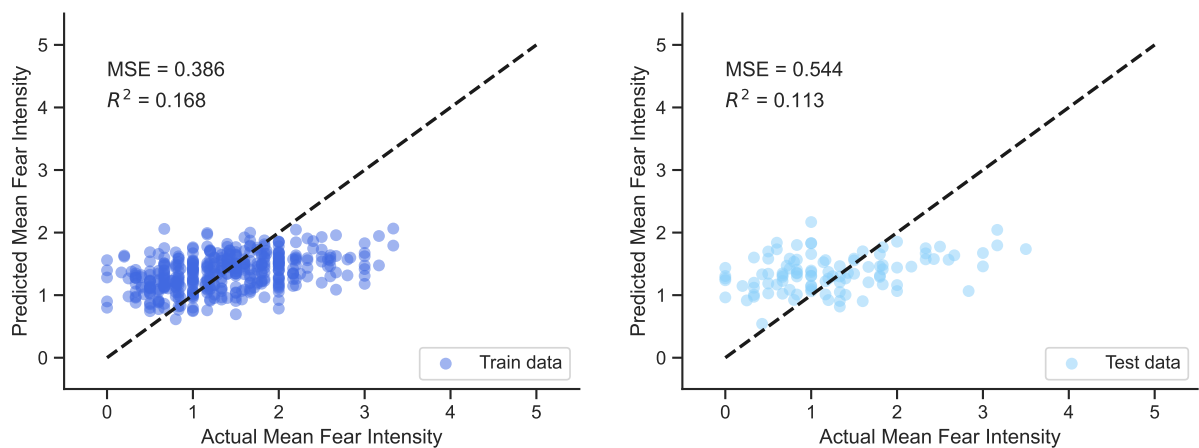


Figure G.25: Scatterplots of the linear model trained with X_{gmm} for fear intensity

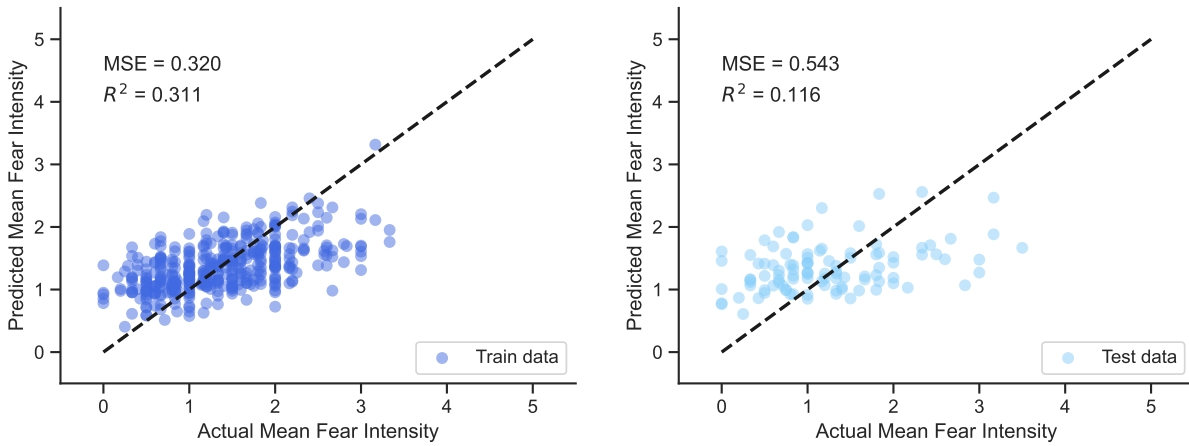


Figure G.26: Scatterplots of the linear model trained with X_{gmm_int} for fear intensity

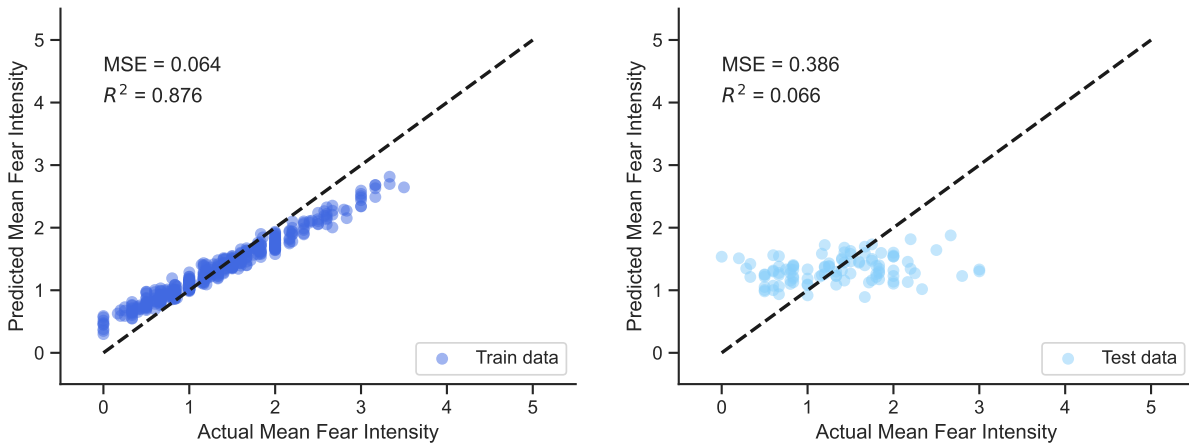


Figure G.27: Scatterplots of the random forest model trained with X for fear intensity

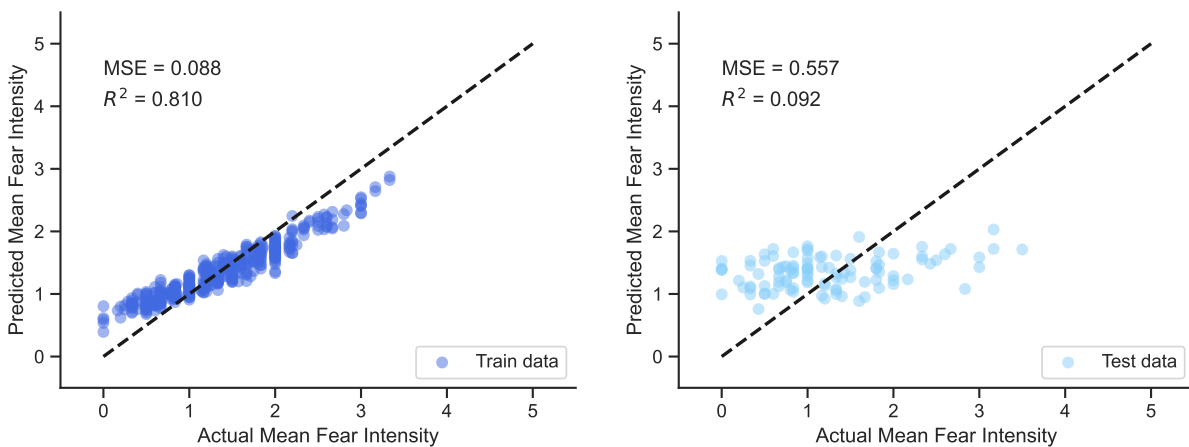


Figure G.28: Scatterplots of the random forest model trained with X_{int} for fear intensity

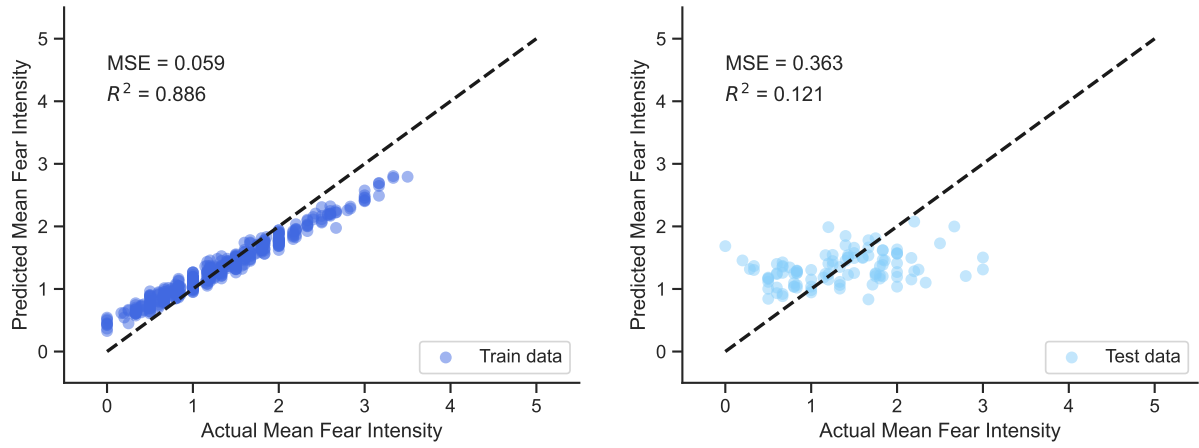


Figure G.29: Scatterplots of the random forest model trained with X_{gmm} for fear intensity

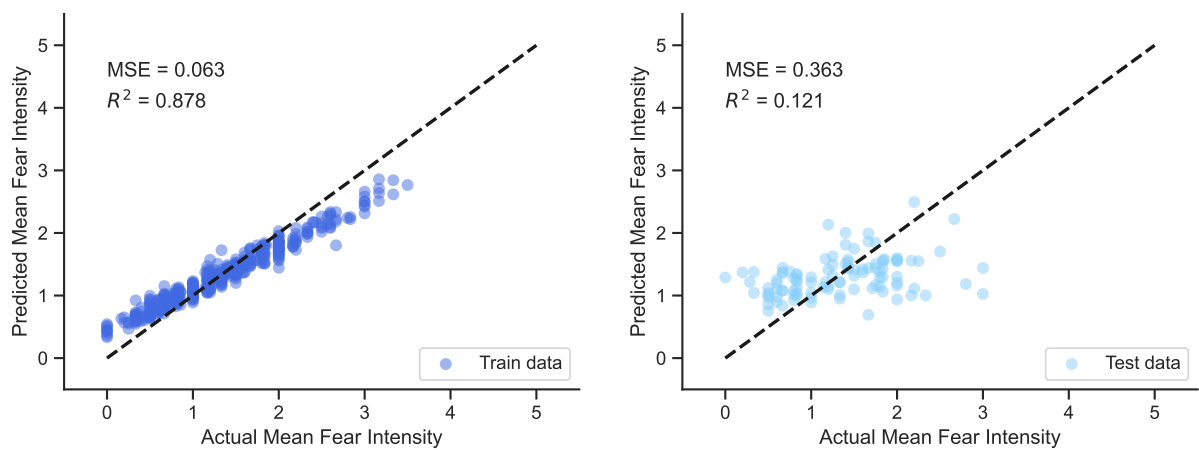


Figure G.30: Scatterplots of the random forest model trained with X_{gmm_int} for fear intensity

G.3.4. Anger Intensity Performance Scatter Plots

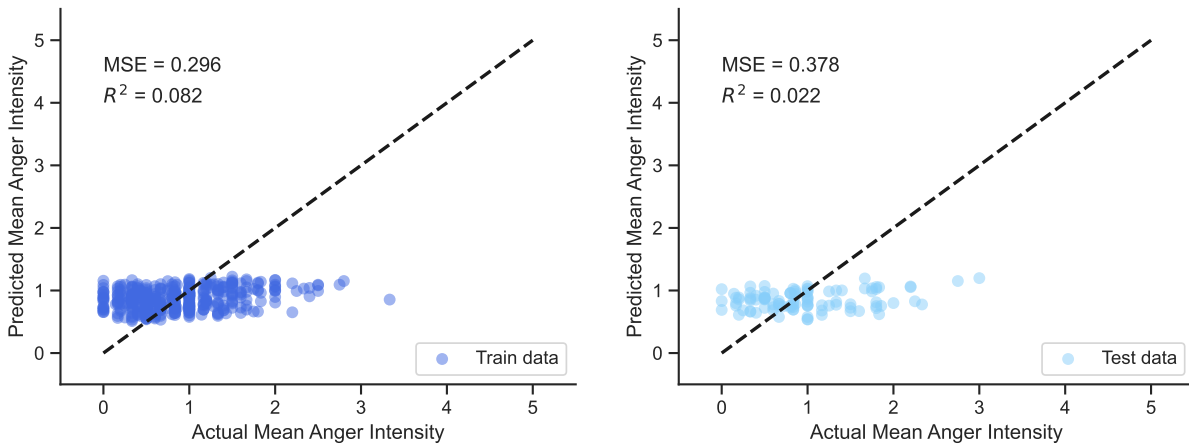


Figure G.31: Scatterplots of the linear model trained with x for anger intensity

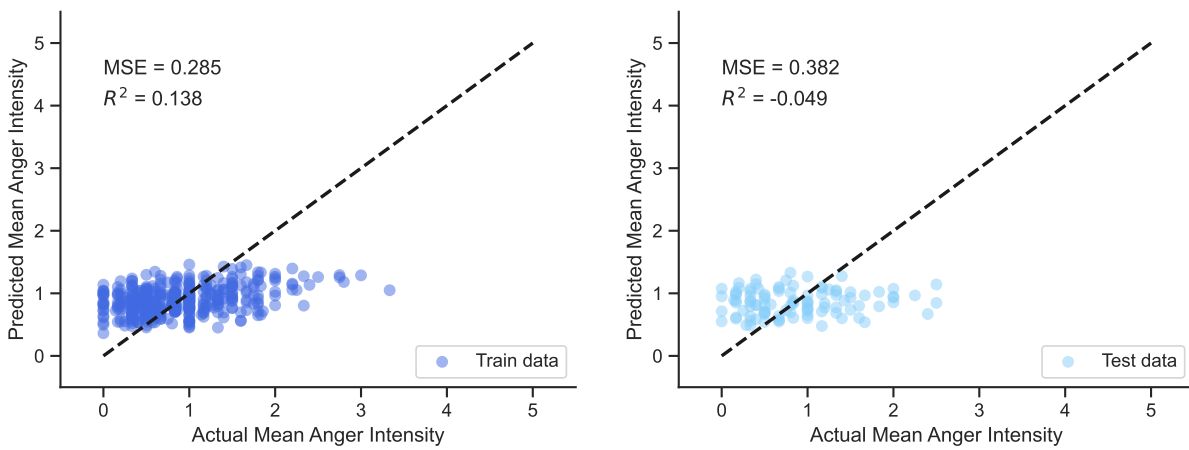


Figure G.32: Scatterplots of the linear model trained with X_{int} for anger intensity

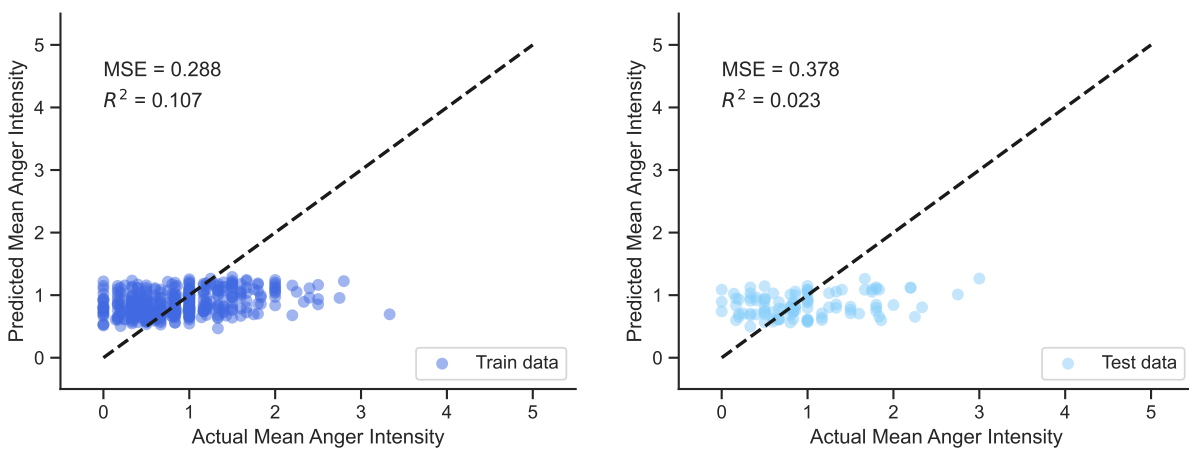


Figure G.33: Scatterplots of the linear model trained with X_{gmm} for anger intensity

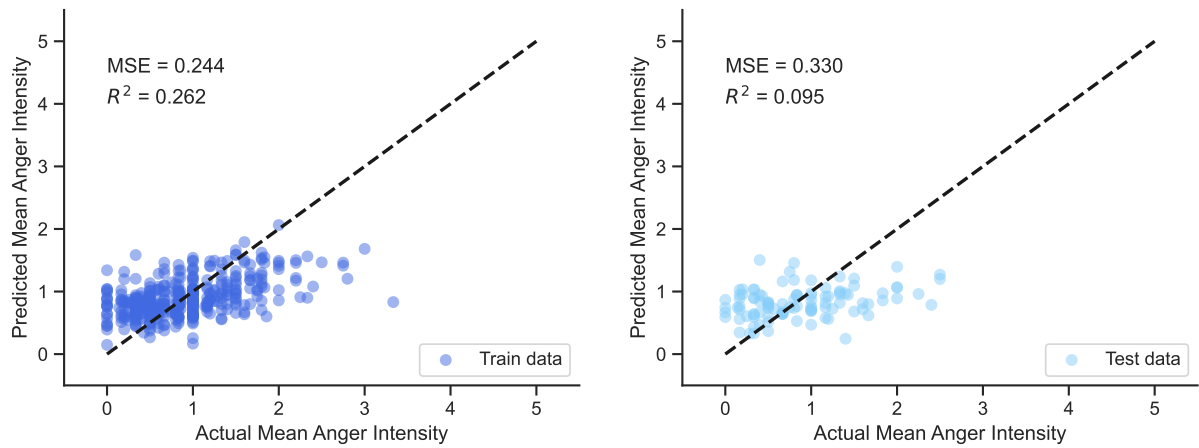


Figure G.34: Scatterplots of the linear model trained with X_{gmm_int} for anger intensity

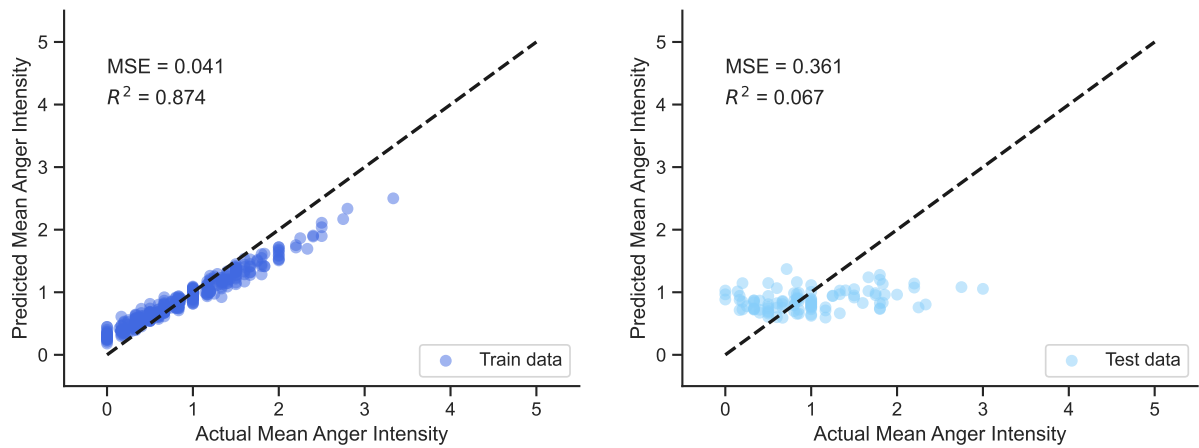


Figure G.35: Scatterplots of the random forest model trained with X for anger intensity

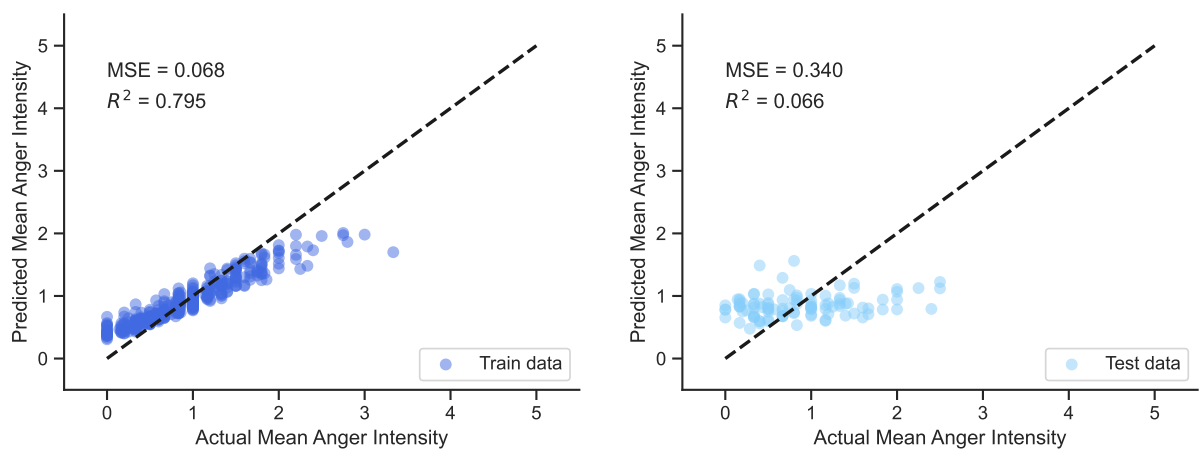


Figure G.36: Scatterplots of the random forest model trained with X_{int} for anger intensity

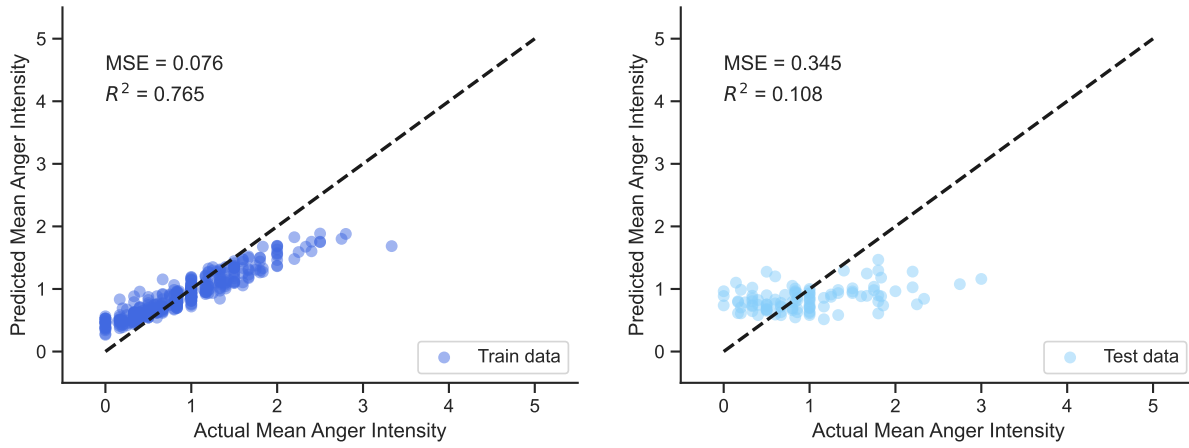


Figure G.37: Scatterplots of the random forest model trained with X_{gmm} for anger intensity

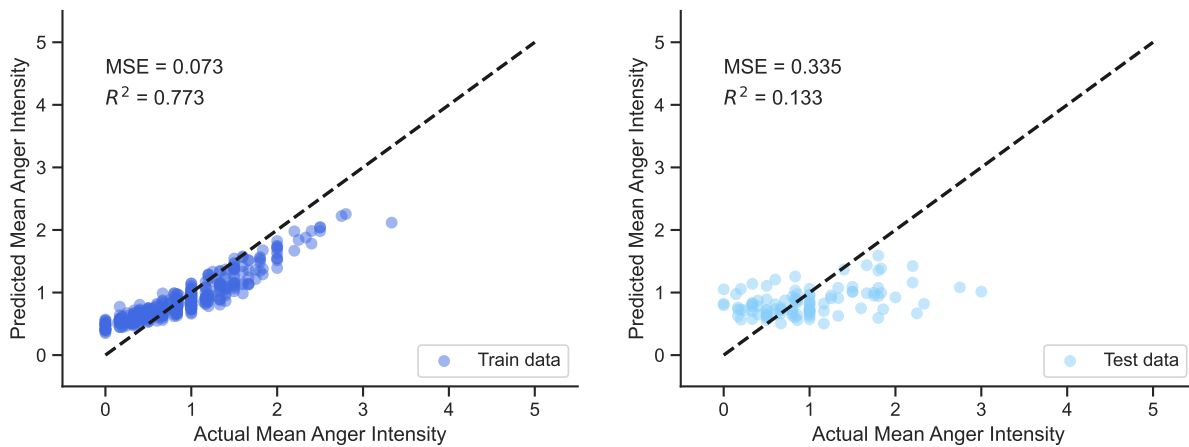


Figure G.38: Scatterplots of the random forest model trained with X_{gmm_int} for anger intensity

G.3.5. Disgust Intensity Performance Scatter Plots

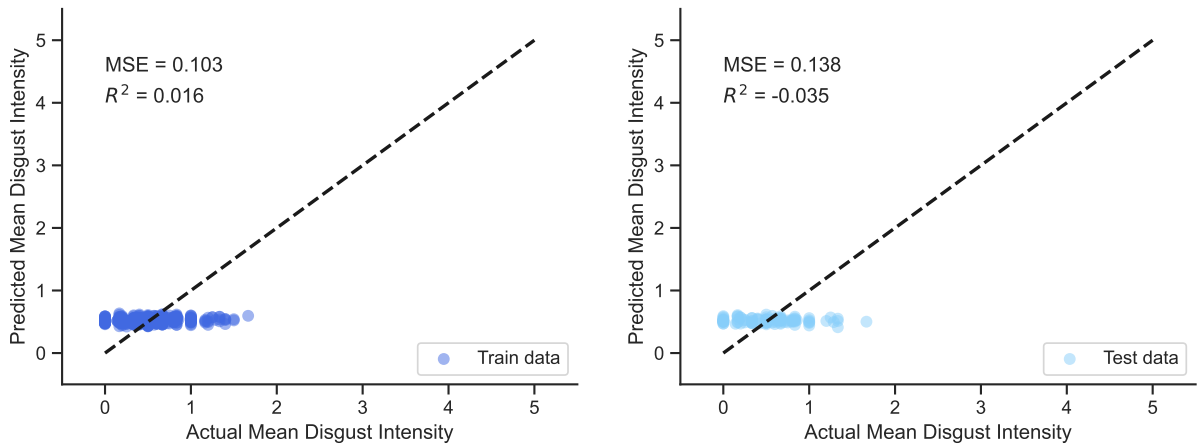


Figure G.39: Scatterplots of the linear model trained with x for disgust intensity

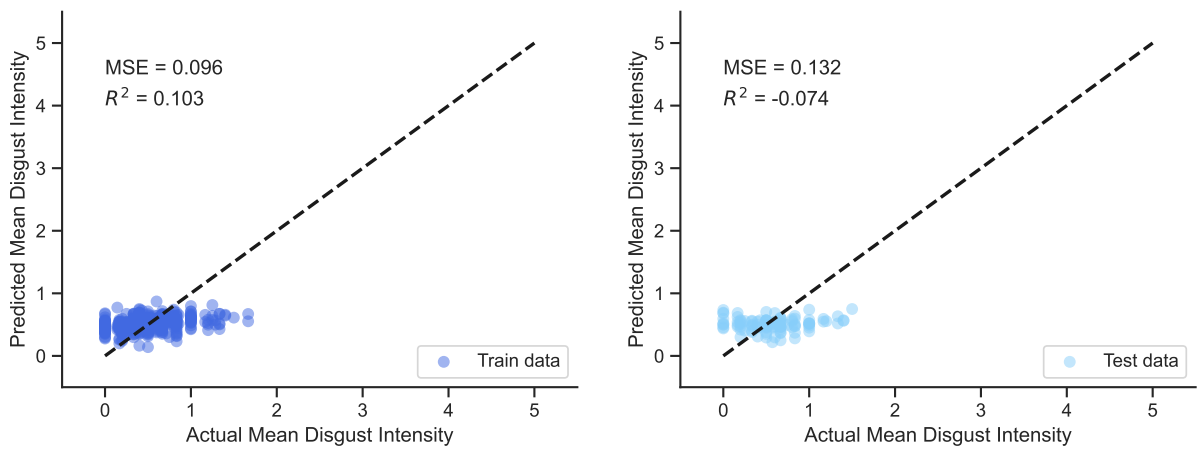


Figure G.40: Scatterplots of the linear model trained with X_{int} for disgust intensity

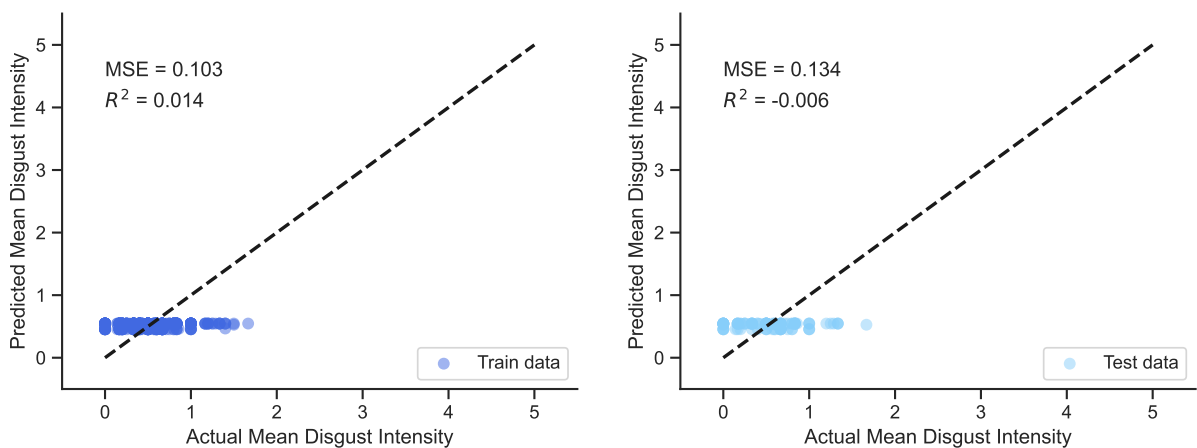


Figure G.41: Scatterplots of the linear model trained with X_{gmm} for disgust intensity

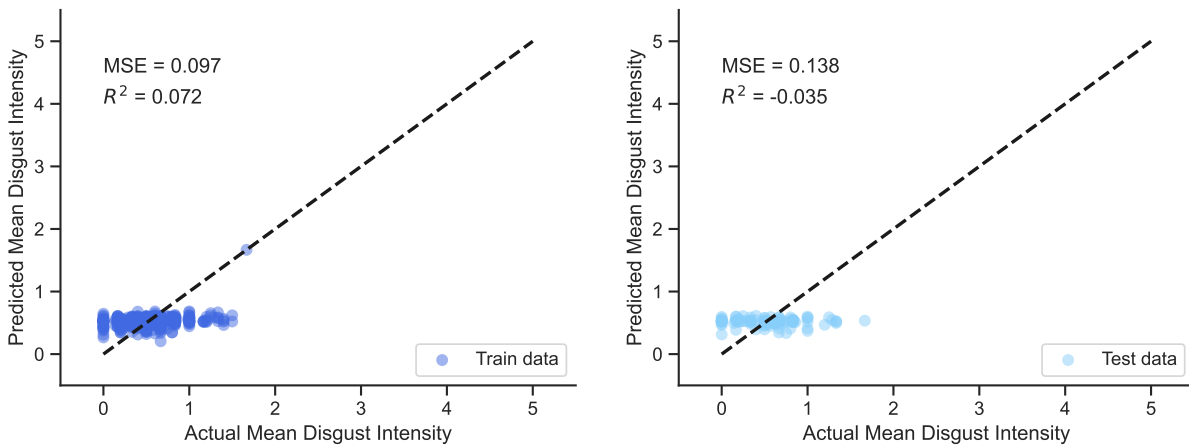


Figure G.42: Scatterplots of the linear model trained with X_{gmm_int} for disgust intensity

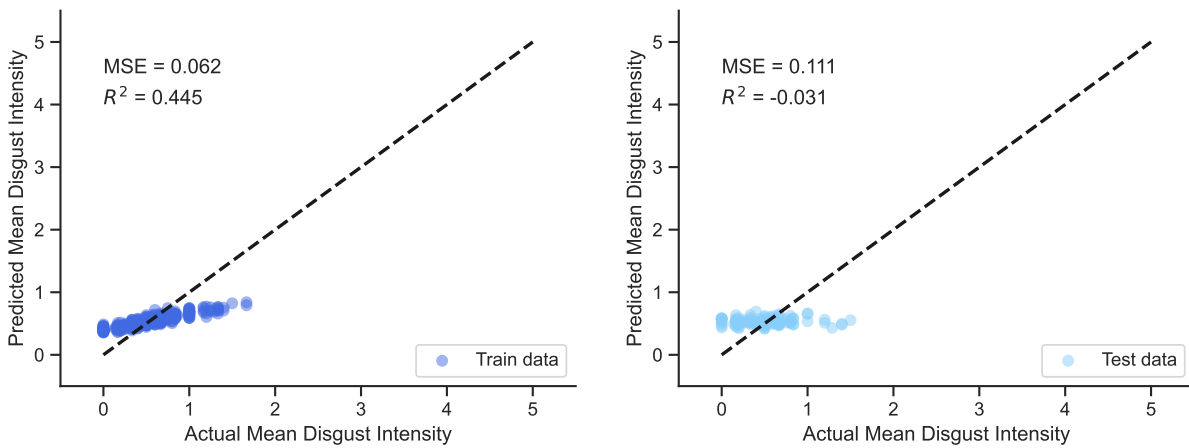


Figure G.43: Scatterplots of the random forest model trained with X for disgust intensity

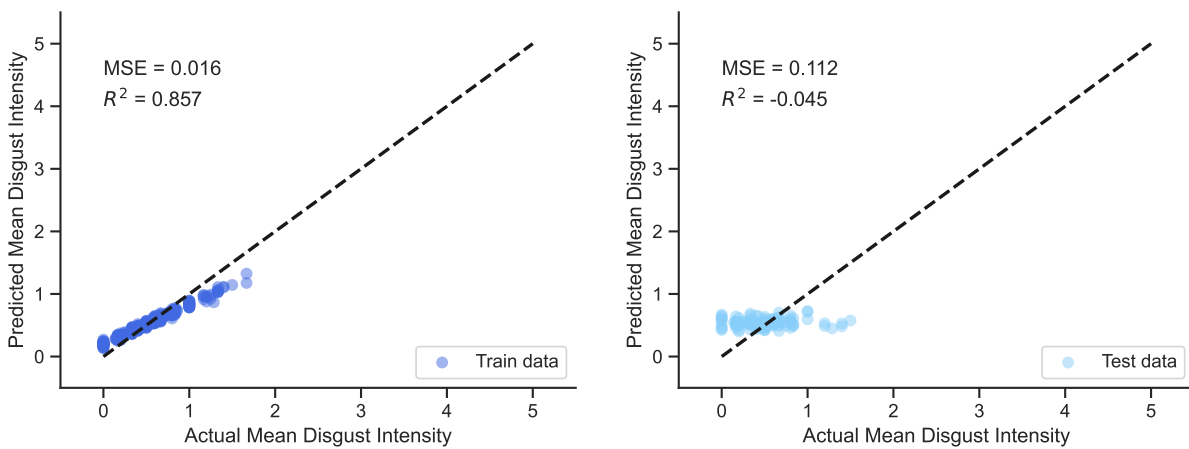


Figure G.44: Scatterplots of the random forest model trained with X_{int} for disgust intensity

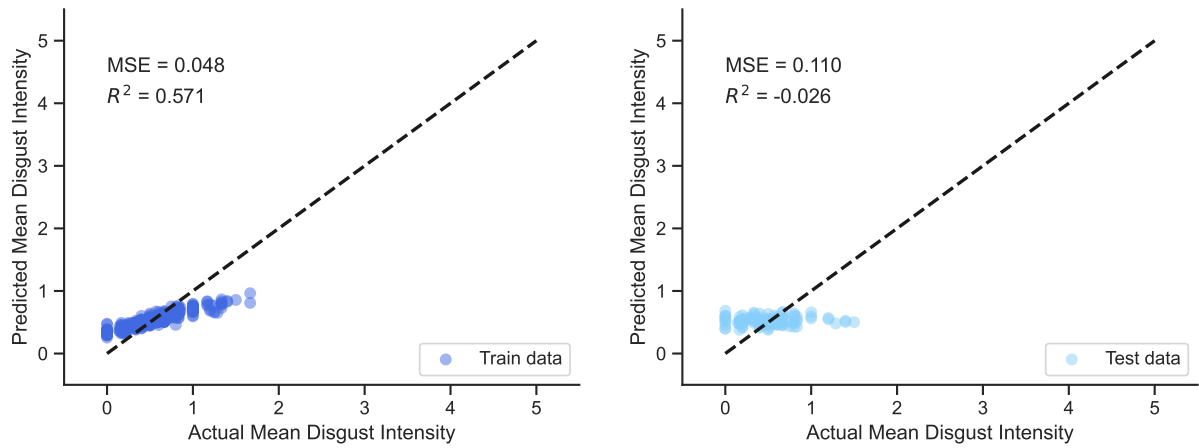


Figure G.45: Scatterplots of the random forest model trained with X_{gmm} for disgust intensity

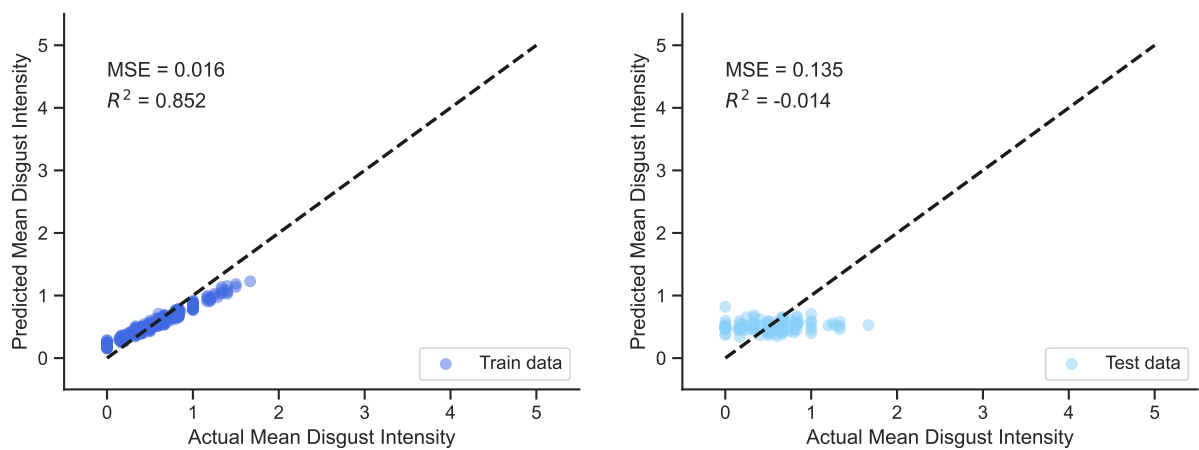


Figure G.46: Scatterplots of the random forest model trained with X_{gmm_int} for disgust intensity

G.3.6. Surprise Intensity Performance Scatter Plots

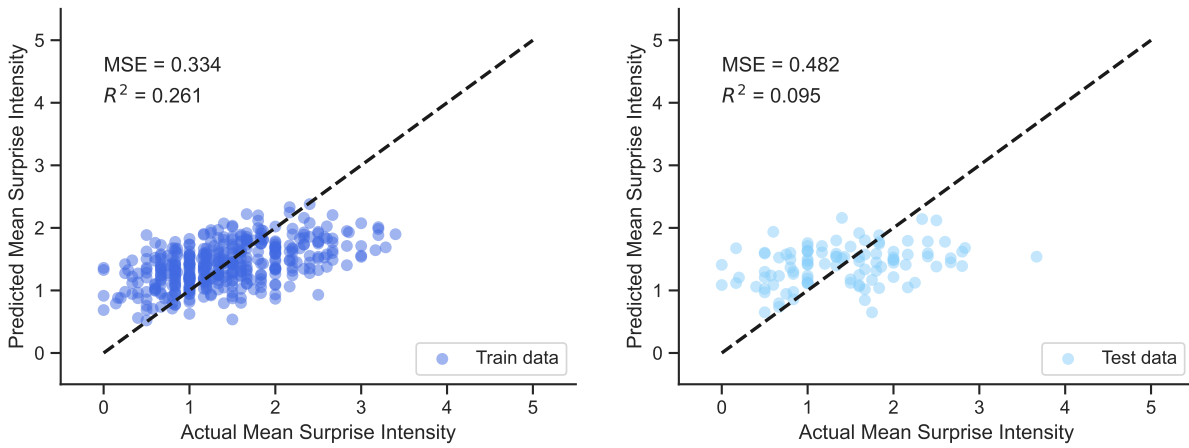


Figure G.47: Scatterplots of the linear model trained with x for surprise intensity

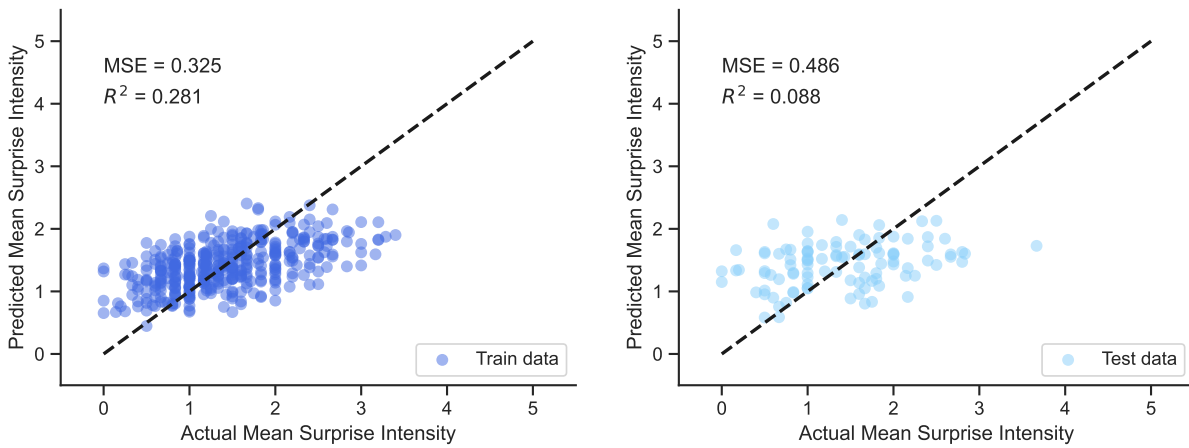


Figure G.48: Scatterplots of the linear model trained with X_{int} for surprise intensity

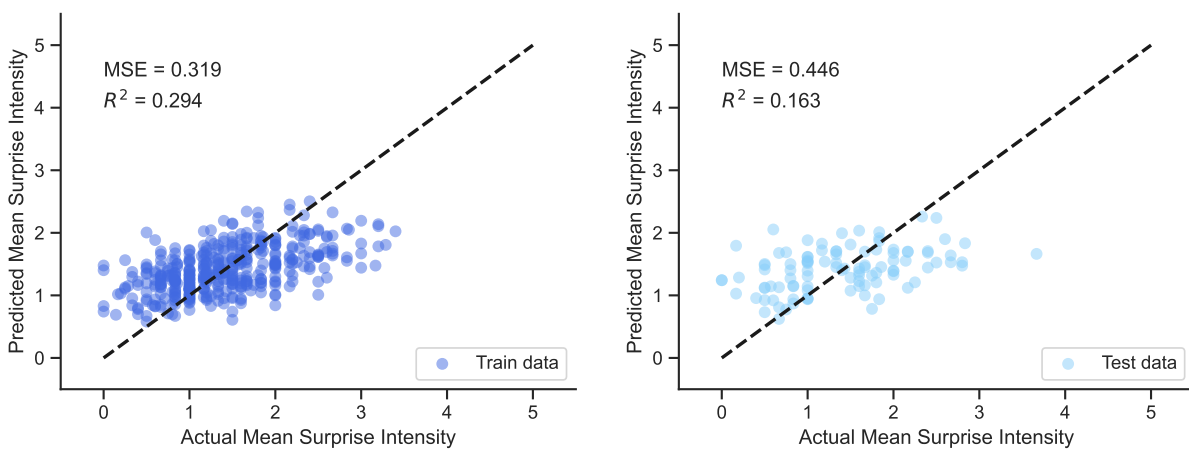


Figure G.49: Scatterplots of the linear model trained with X_{gmm} for surprise intensity

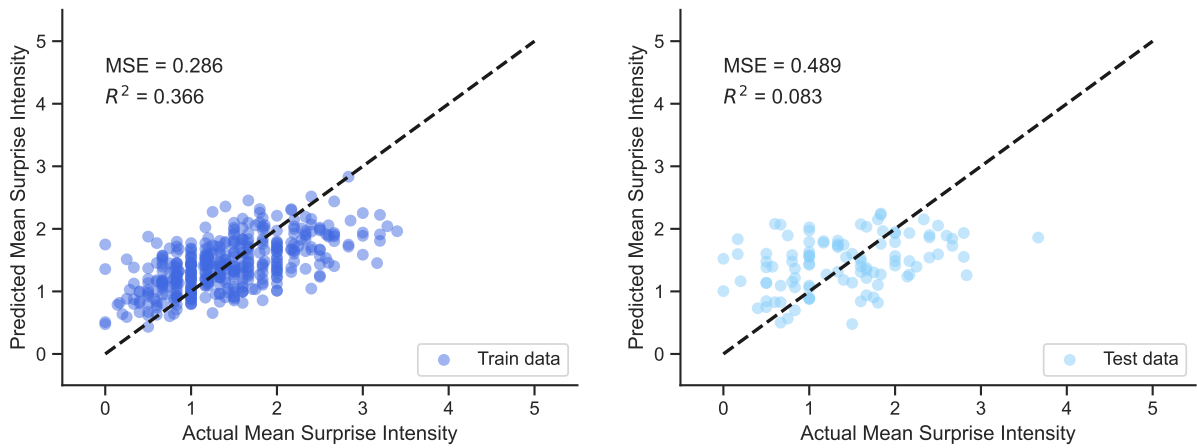


Figure G.50: Scatterplots of the linear model trained with X_{gmm_int} for surprise intensity

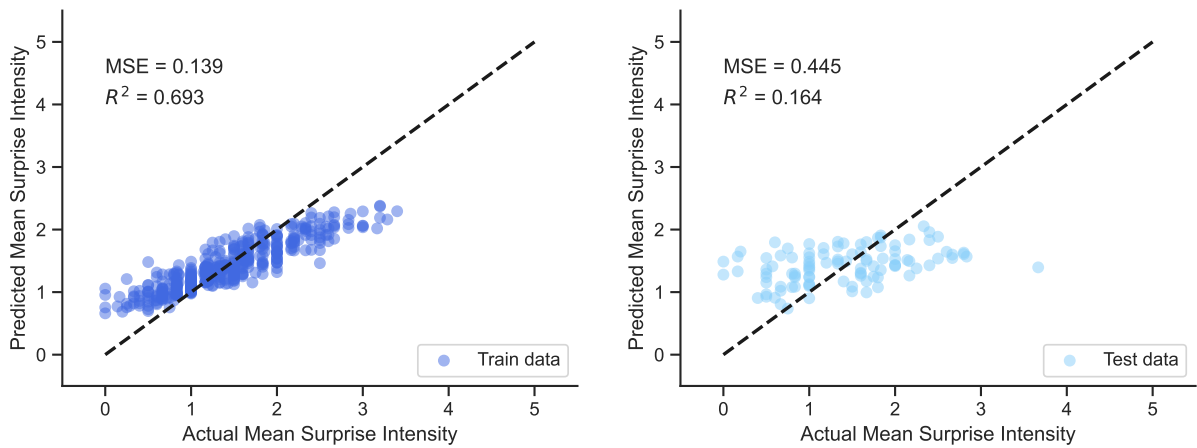


Figure G.51: Scatterplots of the random forest model trained with X for surprise intensity

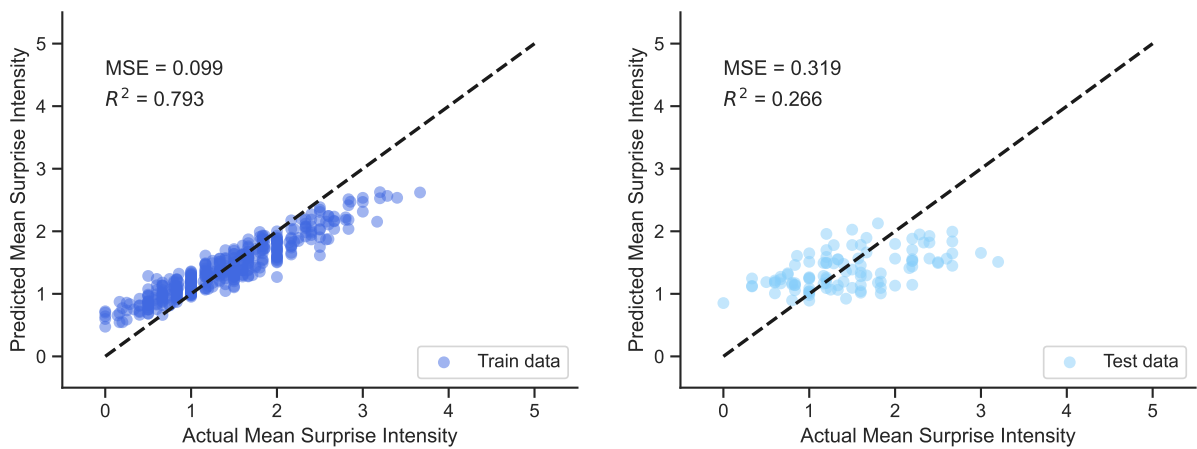


Figure G.52: Scatterplots of the random forest model trained with X_{int} for surprise intensity

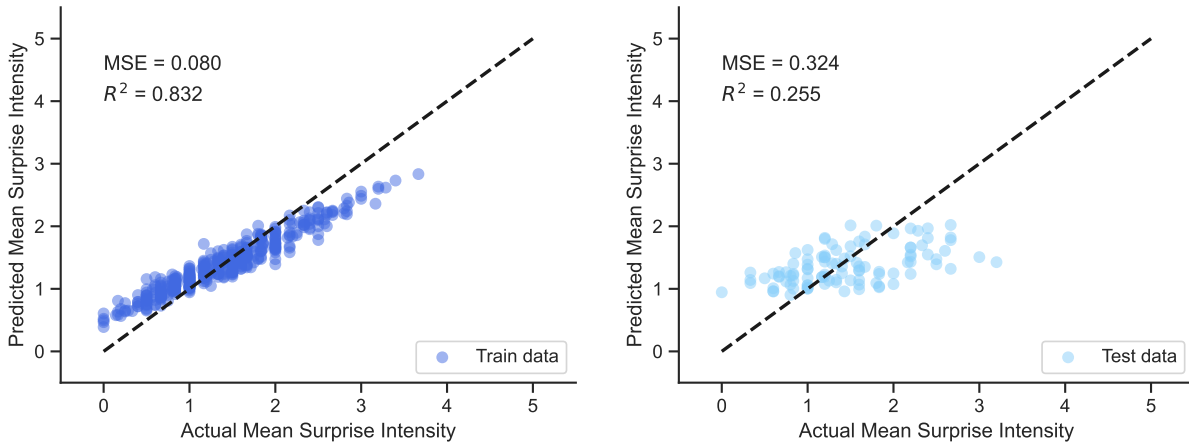


Figure G.53: Scatterplots of the random forest model trained with X_{gmm} for surprise intensity

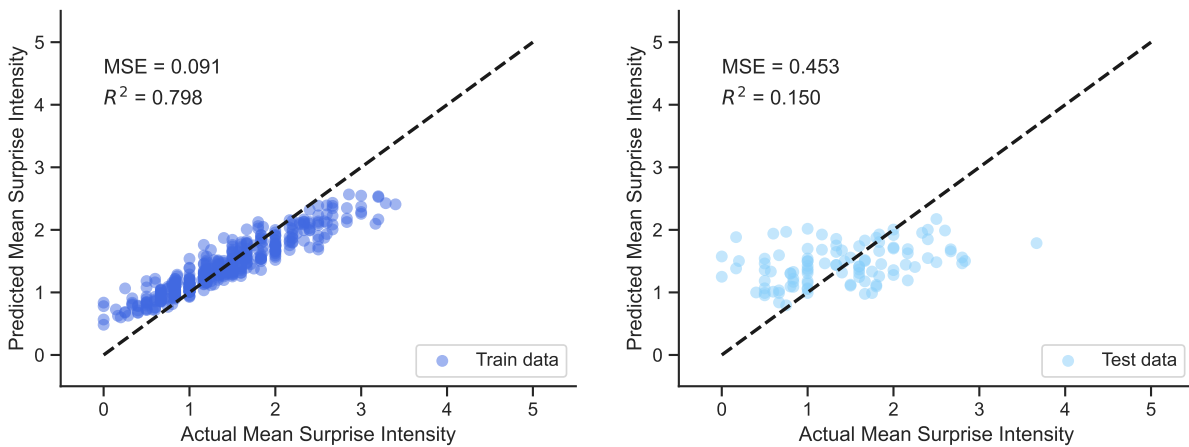


Figure G.54: Scatterplots of the random forest model trained with X_{gmm_int} for surprise intensity

G.3.7. Pleasure Performance Scatter Plots

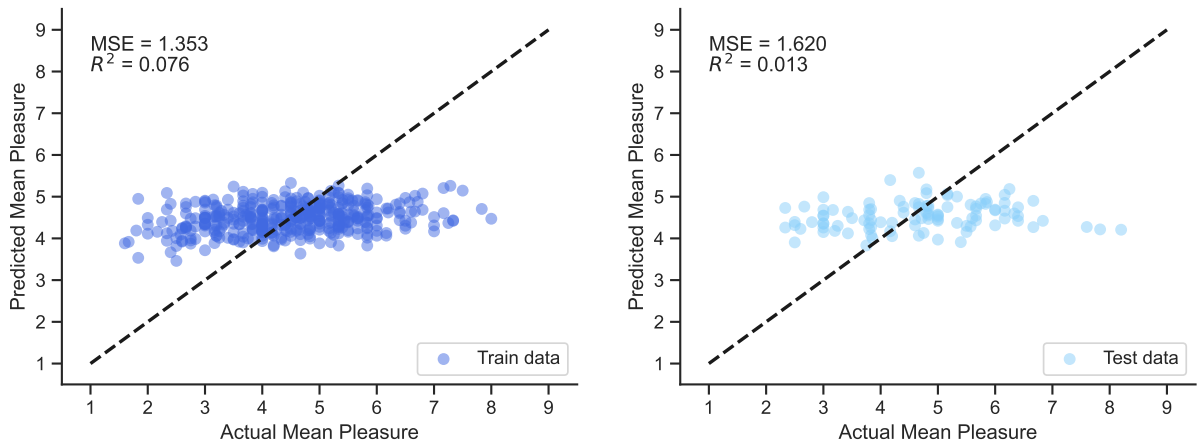


Figure G.55: Scatterplots of the linear model trained with x for pleasure

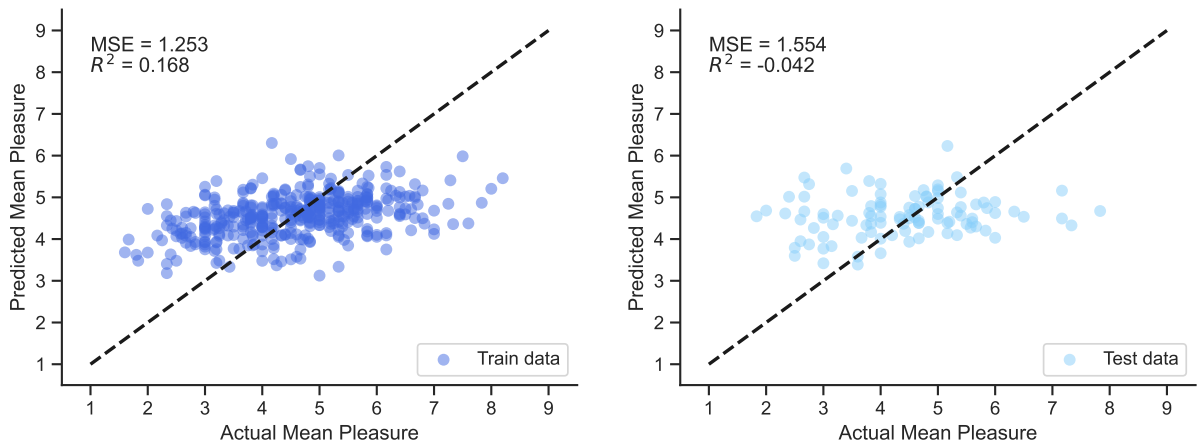


Figure G.56: Scatterplots of the linear model trained with X_{int} for pleasure

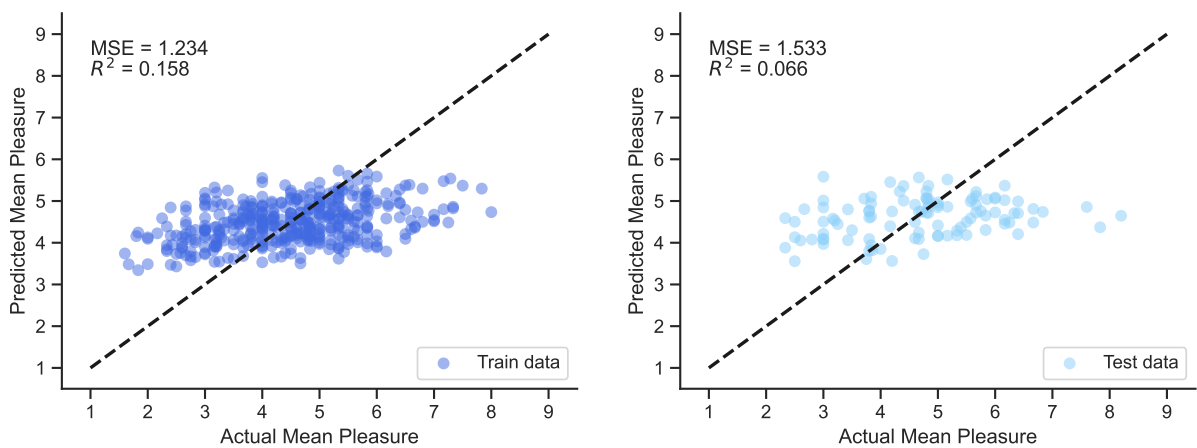


Figure G.57: Scatterplots of the linear model trained with X_{gmm} for pleasure

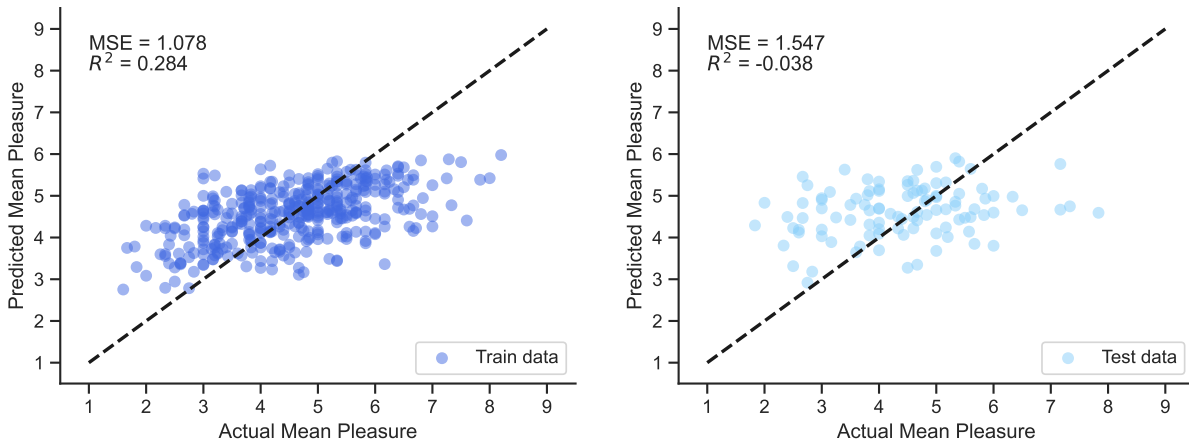


Figure G.58: Scatterplots of the linear model trained with X_{gmm_int} for pleasure

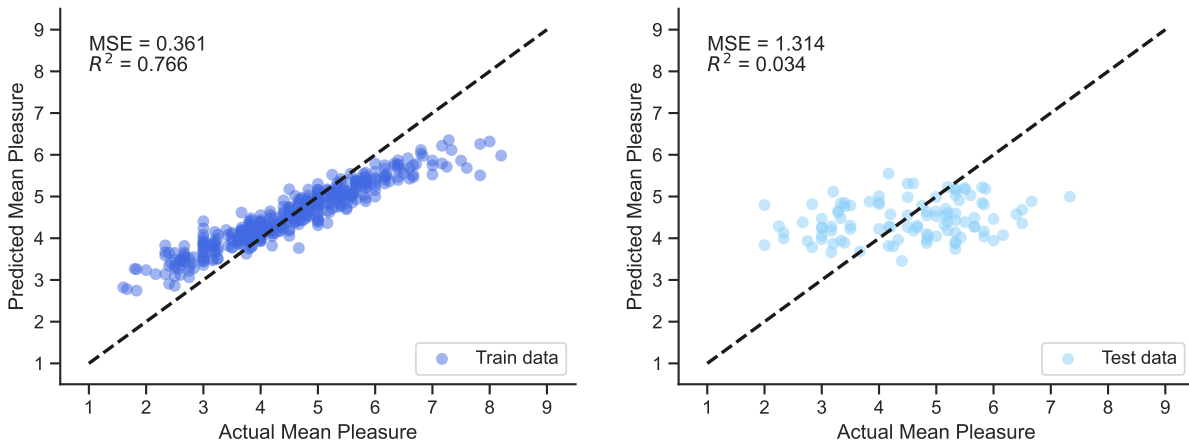


Figure G.59: Scatterplots of the random forest model trained with X for pleasure

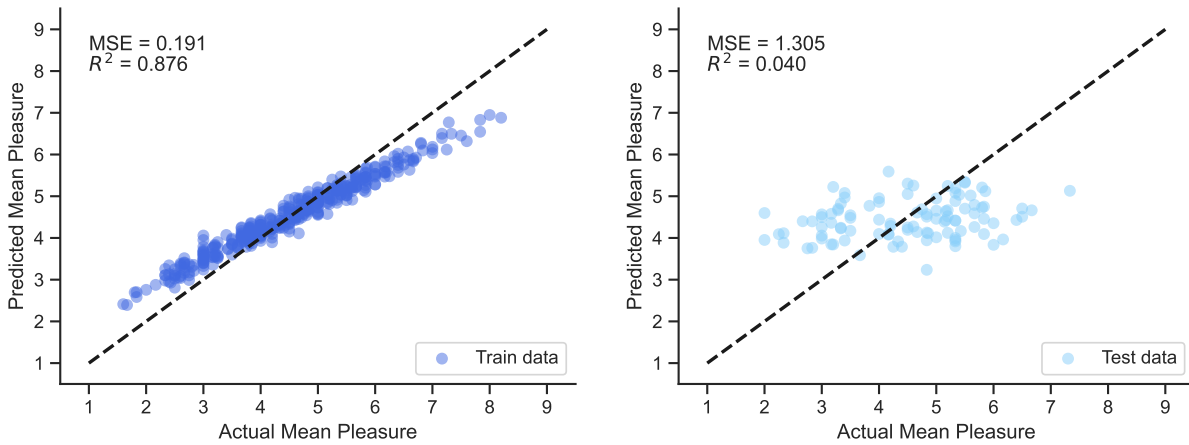


Figure G.60: Scatterplots of the random forest model trained with X_{int} for pleasure

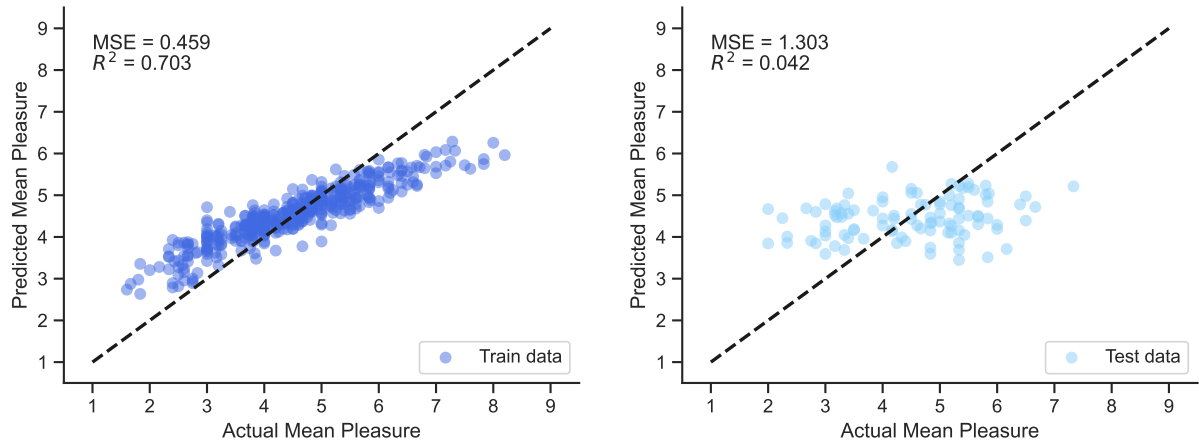


Figure G.61: Scatterplots of the random forest model trained with X_{gmm} for pleasure

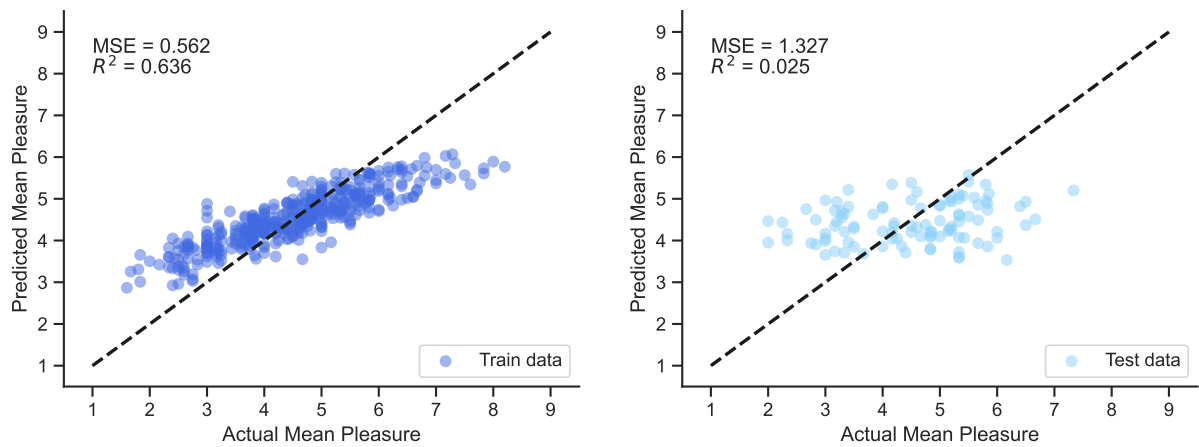


Figure G.62: Scatterplots of the random forest model trained with X_{gmm_int} for pleasure

G.3.8. Arousal Performance Scatter Plots

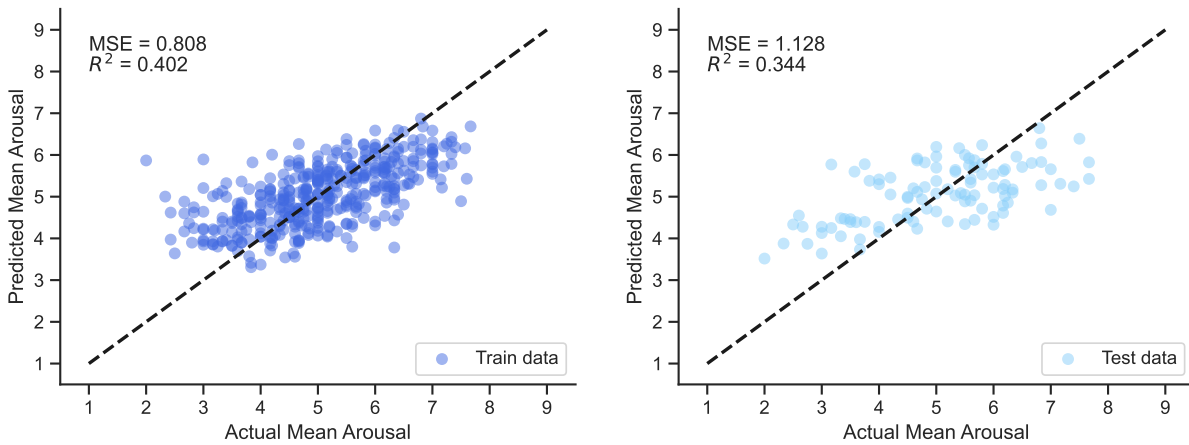


Figure G.63: Scatterplots of the linear model trained with X for arousal

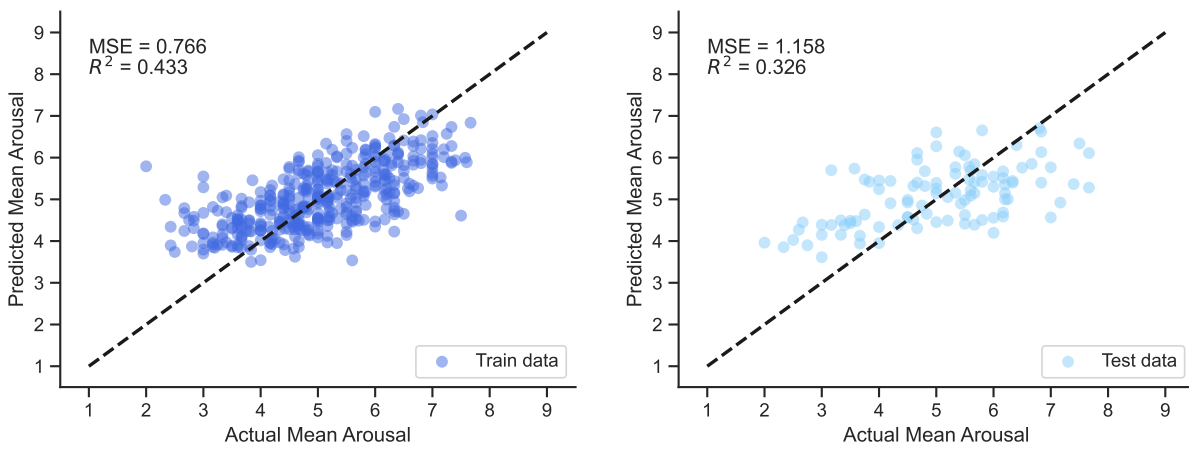


Figure G.64: Scatterplots of the linear model trained with X_{int} for arousal

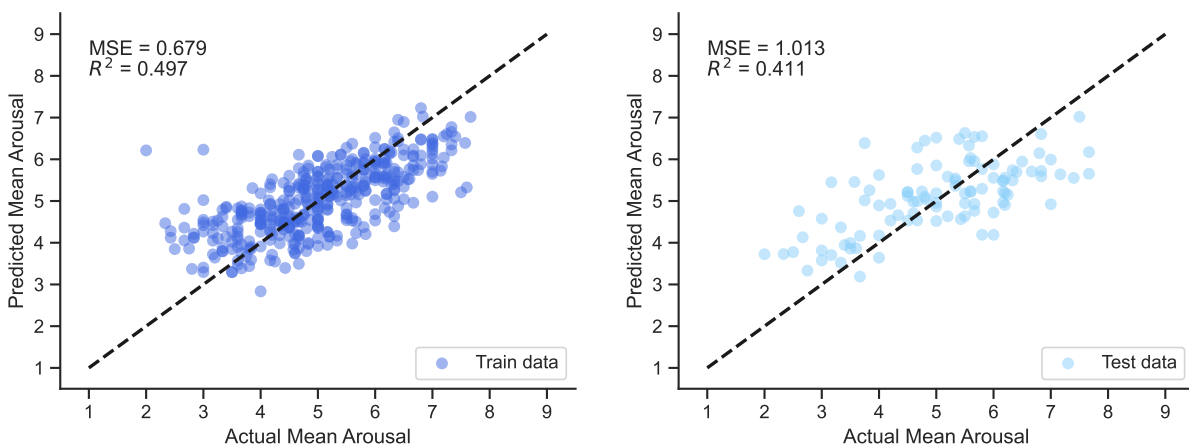


Figure G.65: Scatterplots of the linear model trained with X_{gmm} for arousal

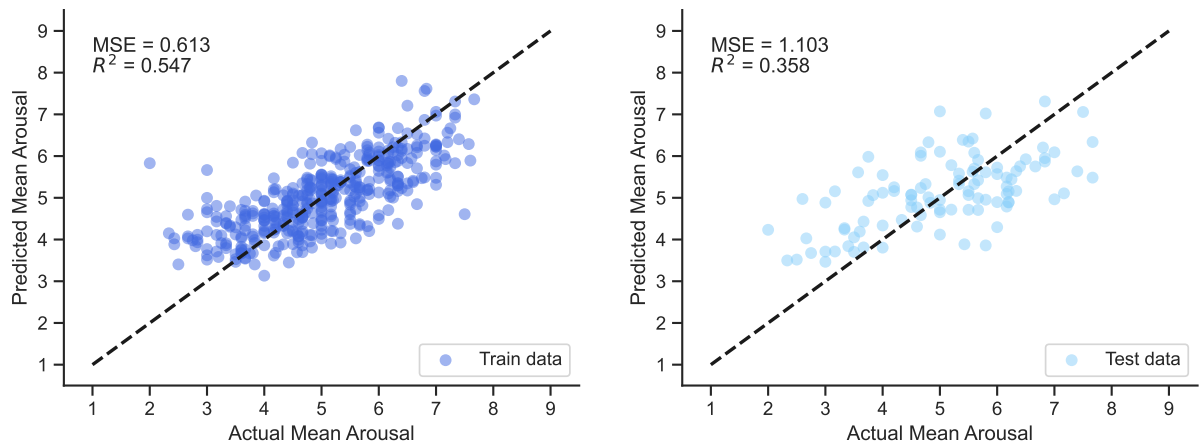


Figure G.66: Scatterplots of the linear model trained with X_{gmm_int} for arousal

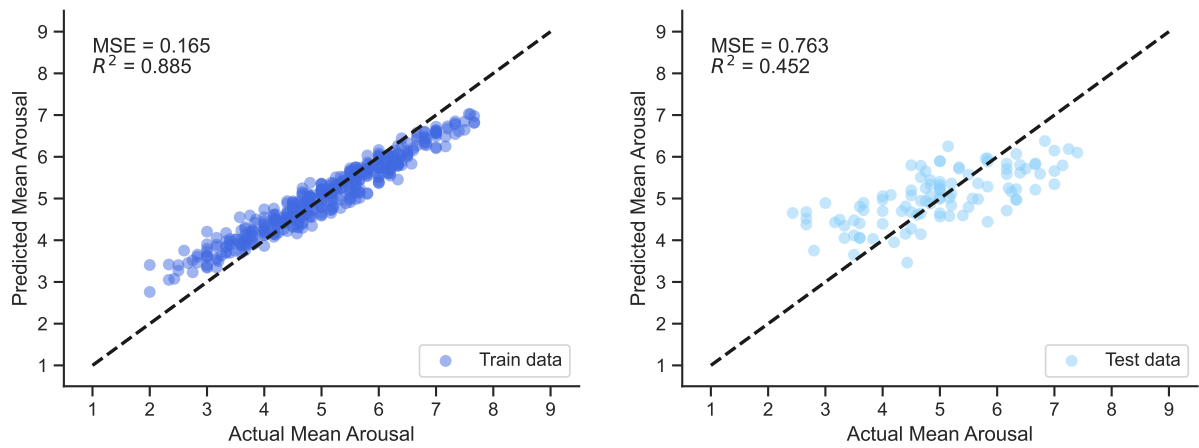


Figure G.67: Scatterplots of the random forest model trained with X for arousal

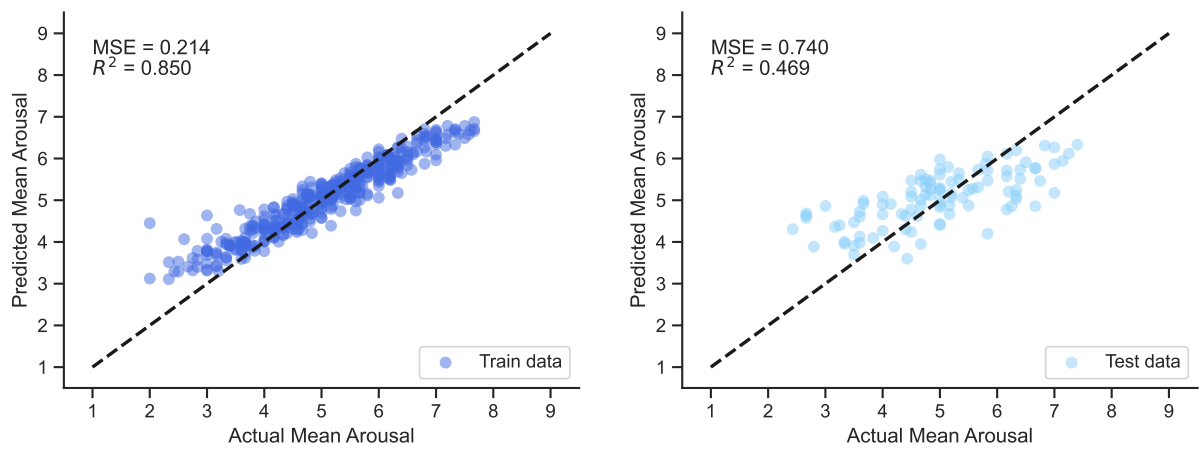


Figure G.68: Scatterplots of the random forest model trained with X_{int} for arousal

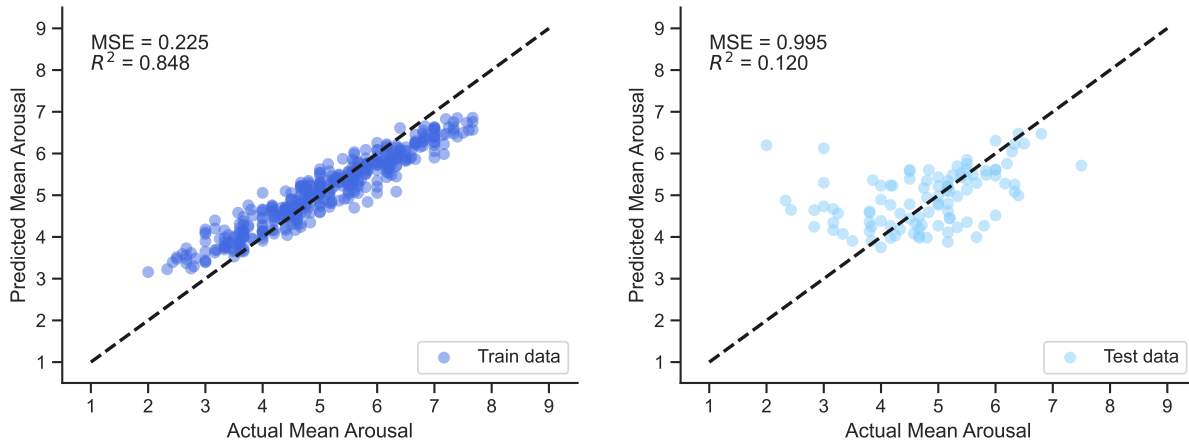


Figure G.69: Scatterplots of the random forest model trained with X_{gmm} for arousal

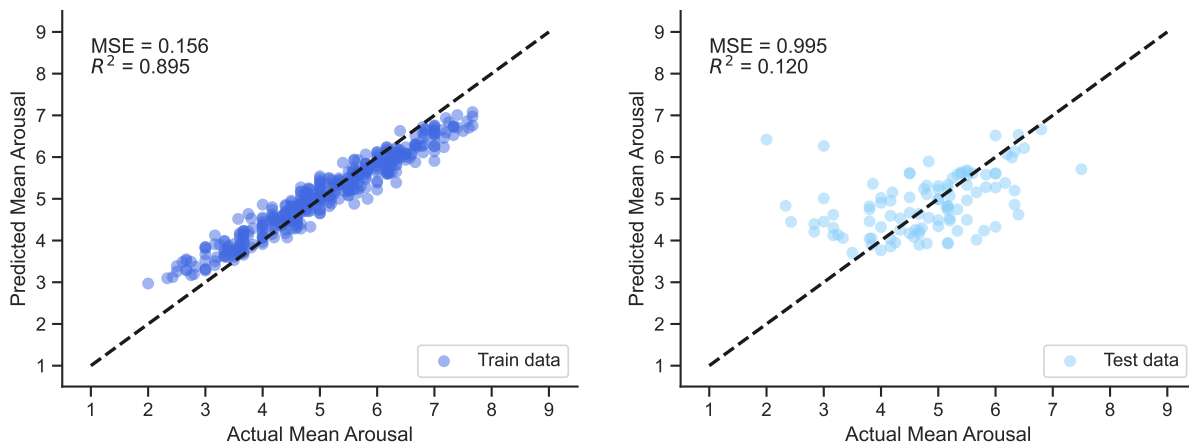


Figure G.70: Scatterplots of the random forest model trained with X_{gmm_int} for arousal

G.3.9. Dominance Performance Scatter Plots

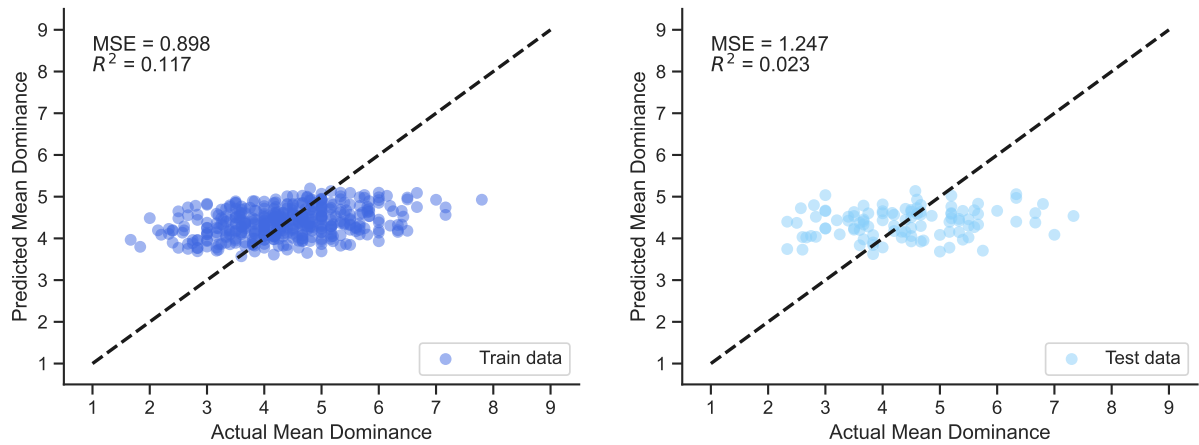


Figure G.71: Scatterplots of the linear model trained with x for dominance

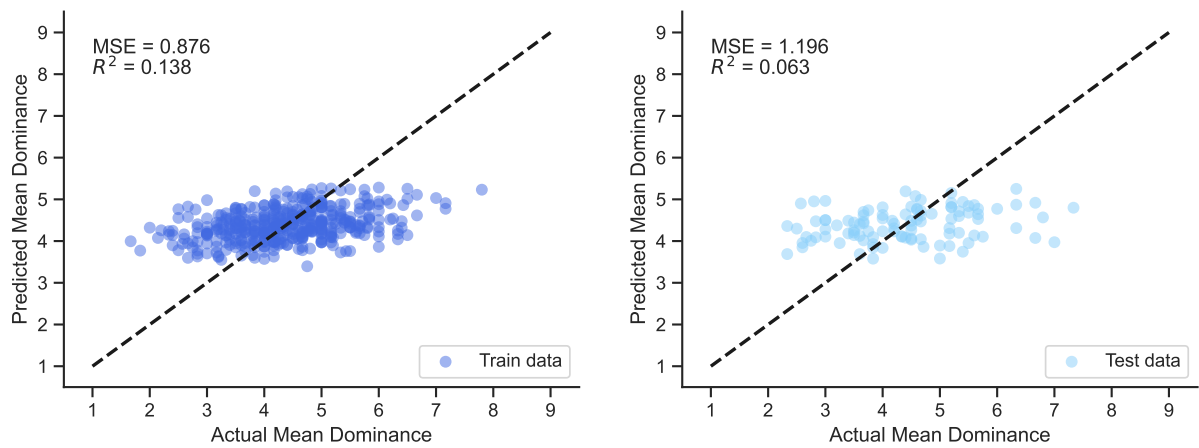


Figure G.72: Scatterplots of the linear model trained with X_{int} for dominance

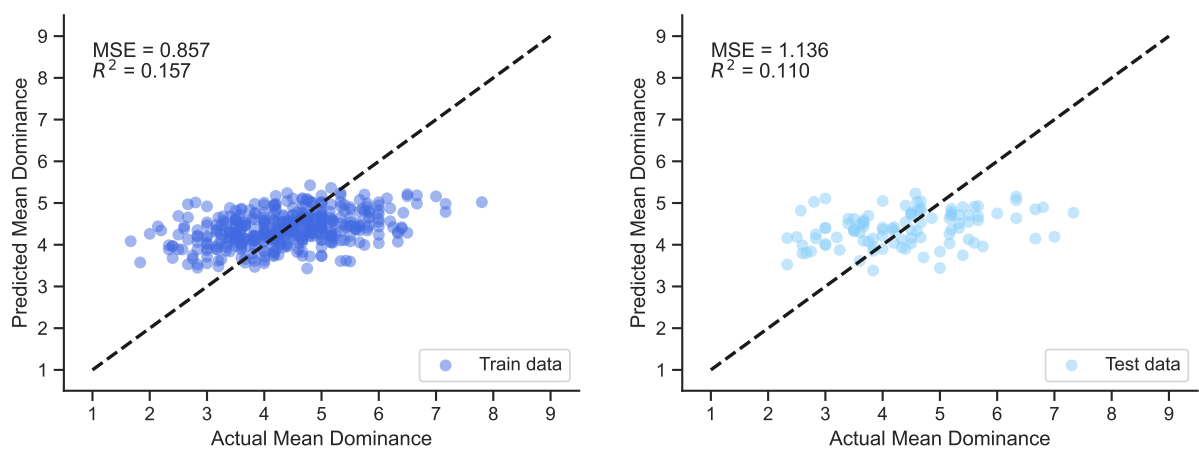


Figure G.73: Scatterplots of the linear model trained with X_{gmm} for dominance

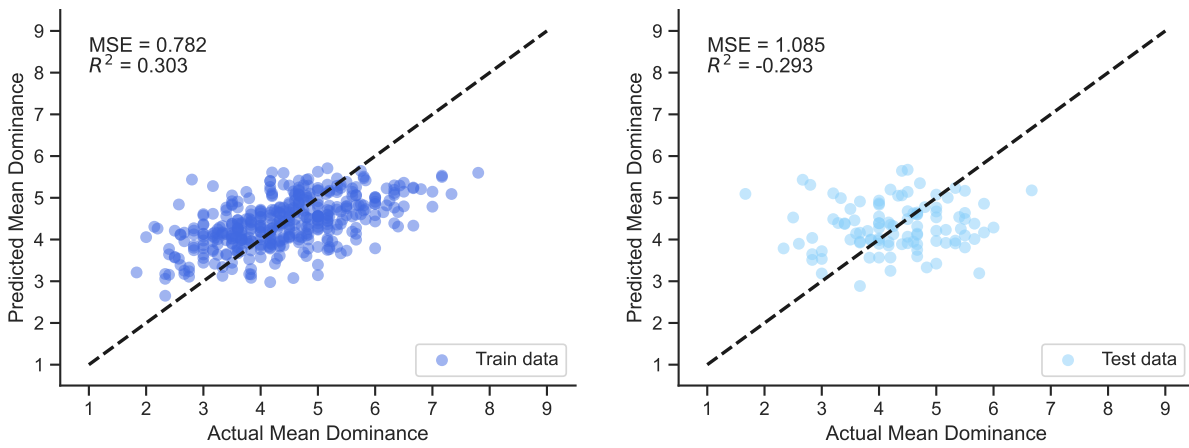


Figure G.74: Scatterplots of the linear model trained with X_{gmm_int} for dominance

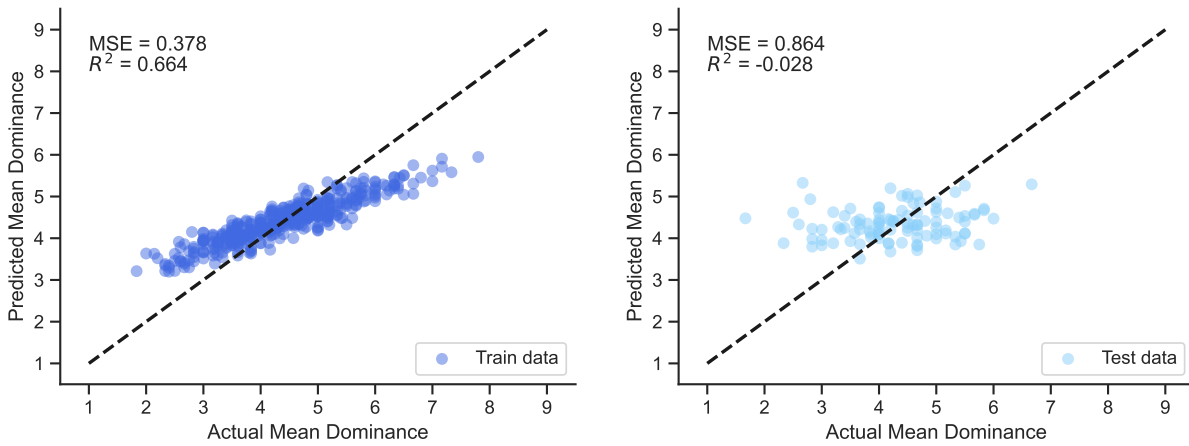


Figure G.75: Scatterplots of the random forest model trained with X for dominance

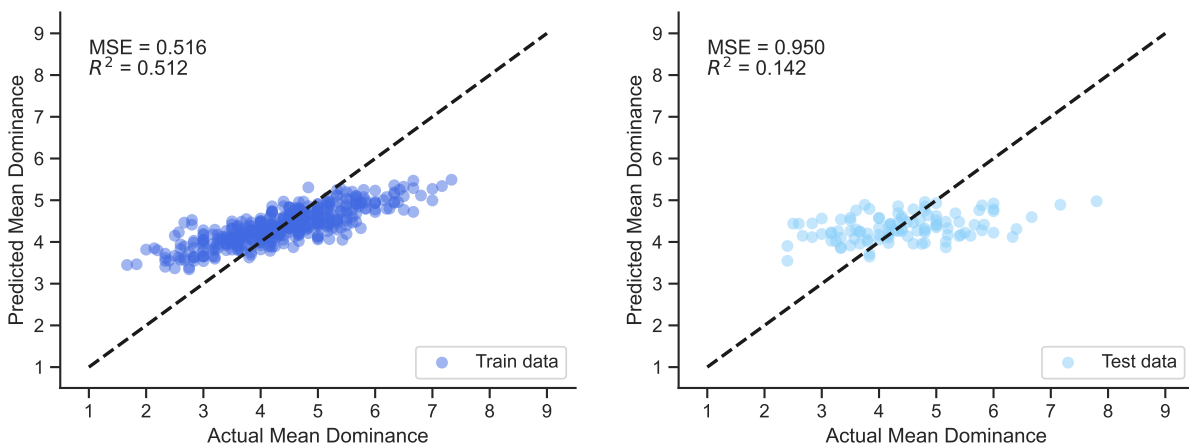


Figure G.76: Scatterplots of the random forest model trained with X_{int} for dominance

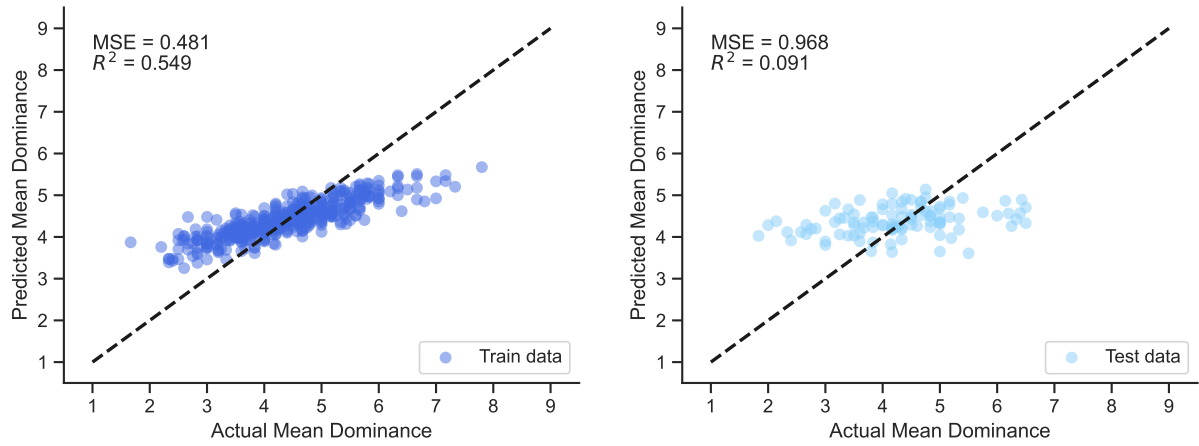


Figure G.77: Scatterplots of the random forest model trained with X_{gmm} for dominance

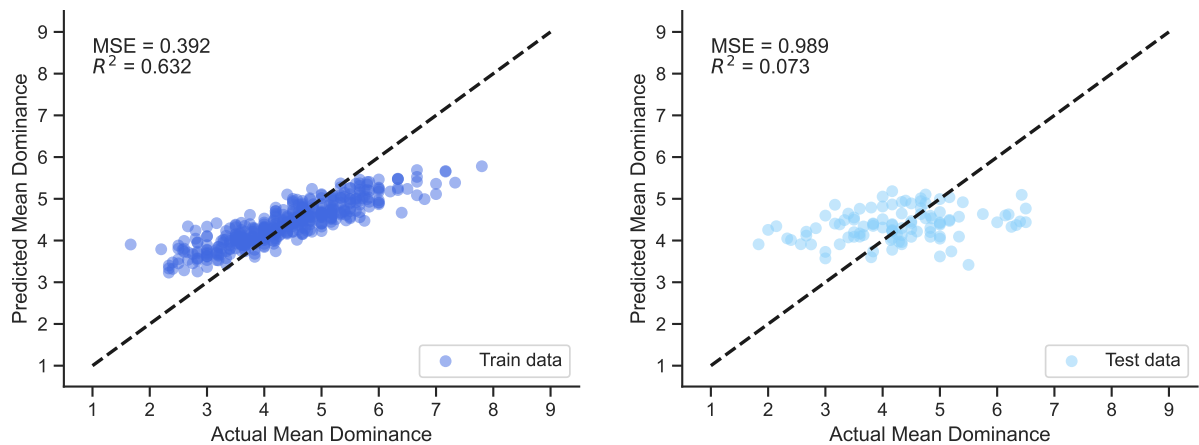


Figure G.78: Scatterplots of the random forest model trained with X_{gmm_int} for dominance