Decoding the Robot's Glance: How Robotic Gaze Validity Shapes Human Cognition and Behavior

Devin Kruse

University of Twente

Bachelor's Thesis (15EC) in Psychology

1st Supervisor: Dr Cesco Willemse

2nd Supervisor: Dr Simone Borsci

Word-Count: 14,775

30.06.2025

Table of Contents

Abstract

As humanoid robots move beyond automated tasks towards collaborative and interactive partners in diverse fields such as healthcare or education, humans' innate tendency to implicitly trust robotic decisions can lead to suboptimal and even dangerous consequences. The uncritical reliance on inaccurate robotic cues could override human judgment, potentially causing serious errors, for example, administering the wrong medication based on the robot's misleading gestures or mishandling hazardous materials in a factory. This work investigated how the reliability of a robot's referential gaze, in tasks of varying complexity, affects human-robot interaction. A self-constructed gaze control system for a screen-based robot was incorporated within a classification game, where participants received attentional gaze cues from the robots. These referential cues differed per robot in their reliability, leading to a high-validity robot (Ryan), a low-validity robot (Ivan), and a third neutral robot, which did not execute any referential gaze. Findings indicate that the existence of referential gaze, reliable or not, leads to significantly higher gaze predictability and faster decision-making as participants develop their own interaction strategies. We found that participants manifested a strong cognitive bias to trust and follow the gaze of the high-validity robot, which was similarly preferred in subjective ratings of anthropomorphism, likability, and intelligence.

Introduction

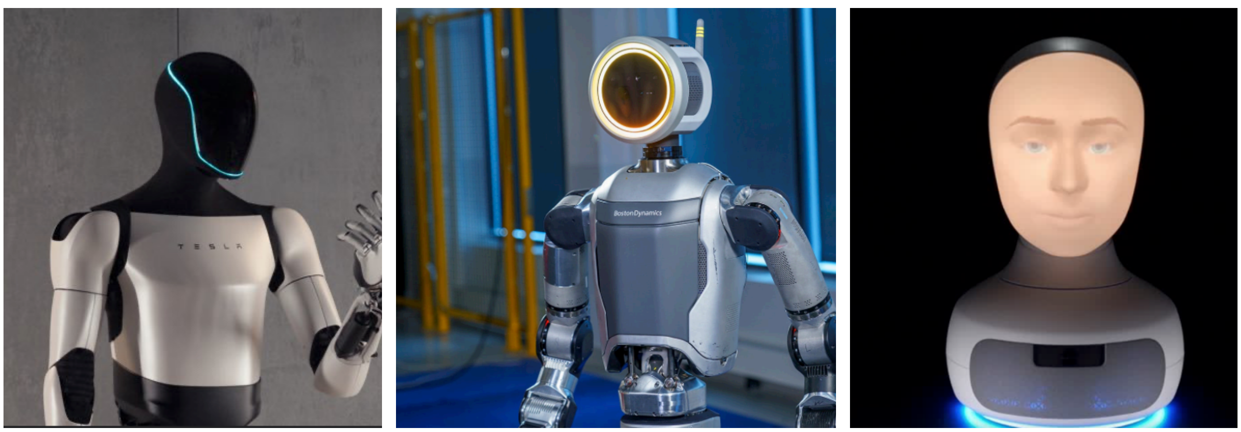
The rapidly changing technological development has empowered machines and autonomous systems to become increasingly adaptive to their environment. Recent breakthroughs in areas such as artificial intelligence and neural networks have led to a transformation in human-machine interaction, far beyond static screens and keyboards. AI-driven systems can interpret speech, detect facial expressions, and navigate in physical environments (Zhou et al., 2023). Leveraging these abilities, AI is fundamentally transforming the field of robotics, enabling systems with advanced autonomy and cognitive functions. Consequently, robotic systems will be increasingly designed with capabilities that extend beyond task execution, emphasizing flexibility, adaptive behavior, and human likeness, particularly in human-robot interaction (Breazeal, 2003).

As robots become an increasingly substantial part of our daily lives, they should arguably behave in ways that feel natural, polite, and predictable to humans, much like how we interact with each other. This social expectation stems from the fact that many of our everyday and professional tasks rely on communication, collaboration, and emotional attunement, requiring robots to behave not only functionally, but also socially appropriately (Breazeal, 2003). Ultimately, a robot's social and emotional awareness contributes to enhanced levels of trust and acceptance (Fong et al., 2003) as well as increased task performance in cooperative settings (Breazeal, 2003). Building on this, research and development in Human-Robot Interaction (HRI) is required to investigate and equip robotics with the necessary verbal and non-verbal capabilities to achieve effective communication.

One such central nonverbal tool that exerts a significant influence on our social consciousness is eye gaze (Kleinke, 1986). However, despite recent advances in this area, many current robots still employ only simplistic or no gaze mechanisms at all (Mishra & Skantze, 2022). As shown in Figure 1, robots like Tesla's 'Optimus' (Tesla, n.d.) or Boston Dynamics' 'Atlas' (Boston Dynamics, n.d.) omit facial features entirely, underlining the technical and conceptual complexities of implementing expressive gaze behavior. To develop machine awareness and implementation, a central, profound investigation of social gaze behavior in human interaction is necessary.

Figure 1.

Examples of Recently Introduced Humanoid Robots.



Note. From left to right: Tesla's Optimus Gen 2, the Atlas robot from Boston Dynamics, and the Furhat robot from Furhat Robotics. Of the robots shown, only Furhat includes an integrated gaze system.

Gaze and Joint Attention in Human Interaction

The human gaze system is a dynamic and active mechanism that constantly interacts with the world. To direct the human gaze, our eyes perform a variety of movements. Two central processes are saccades, referring to rapid jumps between fixation points, and fixations, which define periods of relatively still and stable movements where the brain actively processes visual information (Land & Hayhoe, 2001). Beyond our visual perception, the way and where we look also conveys information about our mental states to others. In fact, gaze serves not only as a perceptual tool but also as a powerful social signal. An individual's gaze conveys information about interest, emotional states, or potential intentions (Emery, 2000). In fact, the ability to follow another's gaze is a fundamental socio-cognitive skill, not only to determine someone's focus but also to enable more advanced social dynamics in human interaction. Joint attention is one of these central building blocks that is highly acknowledged in human-human interaction.

Joint attention is a collaborative, cognitive, and nonverbal process in which two or more individuals share their focus or attention on an external object or activity. What distinguishes joint attention is particularly the mutual understanding that these individuals are attending to something together (Mundy & Newell, 2007). Joint attention can be established when individual A follows the gaze focus of individual B to look at the same object jointly. This process involves two roles: an initiator who directs attention using gaze or gestures, and a responder who follows these cues. In contrast to gaze following, joint attention ensures that both parties are focused on the same object and aware of each other's attention, maintaining a shared focus (Bayliss et al., 2013). Hence, joint attention can be categorized into two components: Responding to Joint Attention (RJA) and Initiating Joint Attention (IJA). RJA describes the role of the responder, who attentively follows the gaze of the initiator. In contrast, IJA describes the role of the initiator, which actively tries to direct the responder's attention towards an object or event (Mundy & Newell, 2007). Eye gaze in particular plays a central role here, acting as a pivotal cue for directing and capturing joint attention. Upon that, joint attention can be achieved through a range of nonverbal behaviors such as head orientation, vocalizations, or pointing gestures (Mehlmann et al., 2014).

To empirically investigate the role of joint attention in human interaction, particularly the mechanisms of initiating and responding to joint attention, scientific research established the gaze-cueing and gaze-leading paradigm. Originating from the Posner cueing paradigm, which investigates the effects of symbolic and reflexive cues on spatial attention (Posner, 1980), the gaze-cueing and gaze-leading paradigm incorporate social stimulus such as eye gaze to provide cues that direct human attention (e.g., Bayliss et al., 2013; Friesen & Kingstone, 1998; Frischen et al., 2007). To study the role of the responder (RJA), a gaze-cueing paradigm is used. In a typical set-up, participants demonstrate faster reaction times while reacting to screen-based targets that align with the cued direction from a human face. Conversely, the gaze-leading paradigm shifts the focus towards the perspective of the gaze initiator, investigating the role of IJA. An examination of this paradigm is more complex, typically involving eye-tracking technology and a reactive screen-based stimulus (usually an avatar or virtual face) to study mechanisms that allow recognition of joint attention (Pfeiffer et al., 2013).

The importance of joint attention in human-human interaction is underlined by its early appearance in development. The ability to execute RJA begins to manifest around 6 to 9 months of age when infants start to follow another person's gaze or pointing gesture. This ability demonstrates infants' early understanding of attentional interest (Mundy & Newell, 2007). Following on that, IJA typically develops between 9 and 12 months, showing a gradual transition from a passive towards an active role in directing another's attention (Tomasello, 1999). In this early stage, IJA is often achieved through gestures like pointing or showing an object. From this early development, IJA manifests itself as a powerful skill, reflecting the desire to share one's interest and exerting influence on another person's mental state (Mundy, 2018).

To further understand the need for joint attention in interpersonal communication, additional insights from research on autism spectrum disorder (ASD) provide a crucial reference point that underscores the critical role of initiating and responding to joint attention in human-human communication (Mundy, 2018). Children with ASD have fundamental difficulties in establishing shared attention, which challenges their ability to navigate through a social world. This impairment creates a cascading effect, significantly hindering the development of social-cognitive skills (Mundy & Newell, 2007). Remarkably, the ability to understand thoughts, beliefs, desires, or emotions – known as the theory of mind – is grounded in the early development of joint attention (Charman, 2000). Consequently, joint attention lays the groundwork for essential social abilities such as sharing experiences and emotions, social bonding, and facilitated turn-taking in interactions (Tomasello, 1999). Serving as a foundational skill in human interaction from an early age (Mundy & Newell, 2007), we argue that joint attention may also be a critical skill in human-robot interaction.

Gaze and Joint Attention in Human-Robot Interaction

Robots that will work and collaborate closely with humans must not only perform programmed tasks, but also be able to understand and participate on a shared social stage (Breazeal, 2003). Joint attention is a primary mechanism that requires the robot's understanding to recognize, respond to, and actively initiate a shared focus with a human partner in real-time (Imai et al., 2003; Scassellati, 2002). The previously discussed concepts of initiating and responding to joint attention are directly applicable, defining the robot's ability to react and trigger such behaviors (Mutlu et al., 2009).

As mentioned above, two paradigms are highly suitable for empirically assessing the concepts of initiating (IJA) and responding (RJA) to joint attention. Similarly, the gaze-cueing paradigm is frequently applied in human-robot interaction, replacing human eyes or faces with robotic cues. In other words, gaze-cueing tasks display the robot's execution of IJA and measure human reaction. In this context, "gaze cueing effects" describe a phenomenon where participants respond faster to a target that appears in a location where the screen-based face is also looking (Willemse et al., 2018). Combining the robot's referential gaze with multiple modalities, such as pointing, creates a more robust and practical effect of IJA (Mehlmann et al., 2014). In addition, robots could exert a "gaze checking" behavior, where the robot looks towards the object and then briefly back to the human face to verify whether the establishment of shared attention was successful (Scassellati, 2002). The robotic capability to direct human attention through IJA has shown significant improvements in task performance, particularly when humans are unfamiliar with the task or situation (Andrist et al., 2017; Pan et al., 2020). On a more subtle level, studies have revealed a powerful effect of IJA in guiding human decision-making as typically shown in gaze-cueing paradigms, where participants' reaction times are usually faster for cued targets (Mutlu et al., 2009; Willemse et al., 2018). Further research has shown that proactive guidance of attention facilitates learning of knowledge and skills as well as task learning and engagement (Kanda et al., 2004).

On the other hand, the "gaze-leading paradigm" refers to the robot displaying a kind of gaze response to the participant's gaze direction (Willemse et al., 2018). Such tasks effectively reverse the roles of the gaze-cueing paradigm, positioning the human as the initiator of joint attention and the robot as the one responding to it. Scientific research has shown that robots with the ability to respond to human gaze cues are often judged as increasingly competent, intelligent, and socially present (Huang & Thomaz, 2011). Responsiveness fosters a sense of being understood by the robot, directly affecting the social dynamics between the two interaction partners (Mutlu et al., 2009). Further, communication with attentionally responsive robots has been shown to increase task performance and efficiency (Huang & Thomaz, 2011). Thus, a central underlying mechanism lies in the robots' ability to demonstrate an increased understanding of human intentions, making the interaction more intuitive (Mehlmann et al., 2014).

Joint Attention in Gaze Control Systems

Gaze-cueing and gaze-leading paradigms provide valuable insights into the human ability to respond and initiate gaze. However, they are often limited to discrete, reaction-based tasks in controlled settings that do not reflect the context-sensitive and continuous nature of real-world human interaction (Pfeiffer et al., 2013). Gaze control systems (GCS) provide a more comprehensive approach that incorporates perceptual input and dynamic gaze coordination (Admoni & Scassellati, 2017). GCS can be categorized into data-driven and heuristic methods. A data-driven GCS uses machine learning from datasets and adapts behaviors through neural networks or reinforcement learning, while a heuristic GCS is based on predefined rules or logic based on human intuition (Lemaignan et al., 2017; Mishra & Skantze, 2022). The underlying architecture of a gaze control system enables the robot to manage its gaze behavior and react to specific environmental events.

Initial research in the domain of joint attention in gaze control systems found a focus on the robot's execution of responding to joint attention (RJA), which displays a reactive behavior (Hoffmann & Breazeal, 2004; Imai et al., 2003). Programming a robot to follow a human's gaze is technically less complex compared to a robot that engages in autonomous, attentional decision-making (Admoni & Scassellati, 2017). Thus, the dynamic implementation of initiating joint attention (IJA) is more challenging as it requires the robotic system to understand environmental context in real-time and direct the human's attention in a socially meaningful way (Admoni & Scassellati, 2017). However, the use of predefined tasks and scenarios can help reactive systems to simulate a proactive-looking IJA behavior. For example, Pereira et al. (2019) triggered IJA by programming a robot to automatically look at the correct puzzle piece a period before it would give a spoken hint in a dialogue act. Similarly, Mehlmann et al. (2014) used a two-step approach, where a robot would first look at the correct puzzle piece and then immediately follow up with a physical pointing gesture to make the instruction clear. Further research equipped the robot with a "gaze checking" tendency to simulate a check during the IJA process by briefly looking at the participant, which made the robot seem increasingly engaging and natural to the participants (Huang & Thomaz, 2011). While such implementation of IJA can also be described as reaction-based behavior, more recent research has focused on building a planned-based architecture, where the robot plans its referential gaze for a future rolling time window rather than being purely reactive. The robot with the planning-based structure was significantly preferred and rated as more interpretable (Mishra & Skantze, 2022).

Despite the above-mentioned advances in gaze control systems, research in this area of joint attention is limited (Admoni & Scassellati, 2017; Lemaignan et al., 2017). Unlike isolated gaze cues, joint attention and social mechanisms require tight temporal gaze coordination and accountability of perceptual input, intention inference, or multimodal expression (Admoni & Scassellati, 2017). Hence, a detailed review of existing studies is essential to validate and investigate the role of joint attention in human-robot interaction.

The implementation of initiating joint attention in gaze control systems was typically designed to be optimally helpful, meaning it consistently directs the participant to the correct objective (e.g., Mehlmann et al., 2014; Mutlu et al., 2009; Pereira et al., 2019). In consequence, such systems would always assume that the robot knows the correct target and executes a correct gaze cue. While foundational, this binary approach – switching IJA as a simple on/off behavior – overlooks a critical aspect of social communication and trust: gaze reliability. In human-human interaction, we do not just evaluate whether a partner provides an attentional cue, but also whether that cue is trustworthy and accurate over time (Frischen et al., 2007). For instance, Bayliss and colleagues (2013) found that participants were faster to reengage with faces that provided a congruent and reliable gaze cue compared to an inaccurate gaze cue, which had cost participants more time monitoring the face. Despite that, frequent HRI literature did not differentiate between reliable and unreliable gaze cues. For example, Pereira et al. (2019) developed a collaborative system in which the robot initiates referential gaze to provide hints for a puzzle. Using a "helper search algorithm", the robot in their manipulated condition always pointed to the correct target, which designed an optimally helpful system. Similarly, Mehlmann and colleagues (2014) investigated the role of referential gaze in a sorting task by comparing accurate gaze cues with no gaze cues in the control condition.

However, a small body of research has begun to address this gap. Research from Admoni et al. (2014) and Staudte and Crocker (2011) explored the complexities of referential gaze reliability. Staudte and Crocker particularly concentrated on the impact of incongruent gaze cues in combination with verbalized statements to investigate speech matching. They found that incongruent gaze cues – where a robot looked at one object while speaking about a different one – significantly disrupted utterance comprehension. Admoni et al. (2014) focused on the effective production of robotic suggestions through the combination of gaze and physical actions. They used incongruent gaze cues to measure compliance and found that a delay between gaze and physical actions significantly increased the likelihood of complying with the robot's suggestion about where to sort a colored block. Despite making significant contributions, the outlined papers demonstrate a predominant focus on referential gaze applied with near-perfect accuracy (e.g., Mehlmann et al., 2014; Pereira et al., 2019). In addition, research that accounted for gaze reliability was primarily focused on measuring social dynamics rather than performance data (Staudte & Crocker, 2011).

A second limitation that has been only partially addressed in the literature is the complexity of the experimental tasks in interaction with the robots (Chen & Barnes, 2014). In more complex situations, people evaluate attentional cues differently than in simple ones (Lavie, 2005). While studies such as Pereira et al. (2019) or Pan et al. (2020) intentionally incorporated varying levels of task difficulty through puzzle complexity or referential ambiguity, many studies utilized a consistent, monotonous level of difficulty (Huang & Thomaz, 2011; Mehlmann et al., 2014; Mutlu et al., 2009). Despite its usefulness in assessing further manipulated variables, such task levels do not account for the complexity of the real world in which humanoid robots will increasingly operate (Admoni & Scassellati, 2017). Table 1 provides an overview of research papers that include the above-discussed variables, such as joint attention mechanisms, gaze reliability, and task complexities.

Table 1.

Overview of Experimental Design Features from Joint Attention Studies that Implemented Gaze Control Systems in Human-Robot Interaction in Comparison to the Current Paper.

Paper	RJA	IJA	Bidirectional Flow [A]	Difficulty Variation [B]	Gaze Reliability Variation [C]
Mutlu et al. (2009)	Yes	Yes	Yes	No	No
Huang & Thomaz (2011)	Yes	Yes	No	No	No
Staudte & Crocker (2011)	No	Yes	No	Yes	Yes
Mehlmann et al. (2014)	Yes	Yes	Yes	No	No
Pereira et al. (2019)	Yes	Yes	Yes	Yes	No
Pan et al. (2020)	Yes	No	No	No	No
The current paper	No	Yes	No	Yes	Yes

Note. IJA refers to Initiating Joint Attention, and RJA to Responding to Joint Attention.
[A] Bidirectional flow means that the experiment directly integrates responding joint attention and initiating joint attention together, without separating the gaze skills. [B] Difficulty Variation is given when the game or task that participants played had different difficulty or complexity levels. [C] Gaze Reliability Variation refers to the fact that robots' gaze behavior differed in terms of pointing to the correct target, for example, also pointed in the wrong direction.

To conclude, research on joint attention mechanisms around human-robot interaction has already highlighted its improved engagement, task performance, and efficiency (e.g., Huang & Thomaz, 2011). However, the rapid development towards adaptive and social robots requires more context-sensitive research, particularly considering a robot's reliability in guiding human attention. While robots have become significantly more intelligent (Breazeal, 2003), we cannot ideally rely on them in every situation, considering that they will play an increasingly responsible role in our everyday lives. As a collaborative and interactive partner across various fields, robots pose significant societal risks, as they can guide human decision-making through misleading gaze cues. The implicit development of complete trust and automation bias (Parasuraman & Manzey, 2010), even to the point of overriding one's judgment, can lead to blind following, resulting in costly errors. For instance, in factories, this could lead to increased costs due to repeated errors. Of greater significance, such automation bias could also appear in the health and care fields, posing dangerous consequences (Goddard et al., 2012). If we equip robots with social intention tools like the initiation of joint attention, research must address the consequences of such decisions.

Thesis Outline

The purpose of this research was to develop a reactive, screen-based gaze control system that enables real-time interaction in a context-based task. In alignment with the research aim, this context should provide the robot with the opportunity to direct the participant's attention using referential gaze. Finally, this mechanism of initiating joint attention to a specific side allowed for control in its reliability and considered gaze quality. Taking this into account, the researcher programmed a classification game with two categories on the right and left side, allowing participants to drag and drop cards into one of the two categories. This involved internal communication with the built gaze control system to enable the robot to plan its gaze. To compare the conditions, three different robots were created, each displaying different gaze behavior. Two of the three robots, "Ryan" (high validity) and "Ivan" (low validity), displayed referential gaze towards one of the classification categories. Conversely, the third one, "Carl" (neutral robot), did not apply any gaze. To account for differences in gaze reliability, “Ryan” belonged to a “high-validity” condition, pointing to the correct classification in 80% of the trials. At the same time, “Ivan” displayed the “low-validity” condition, pointing towards the incorrect side in 80% of the trials. Finally, to account for complexity variation, statements were categorized into easy and hard categories.

Consequently, this work establishes a unique triadic comparison that not only concentrates on the existence of referential gaze but also its quality. Our study design moves beyond simple dichotomies and accounts for task complexity and differing gaze accuracy. The aim of this multifaceted approach is to enable more profound insights into the execution of joint attention, guiding human attention and decision-making in human-robot interaction. The following research question was formulated: **“**Given a varying task complexity, how does the reliable execution of referential gaze impact humans’ cognitive and behavioral processes, particularly their visual strategy and gaze-follow decisions? “

In line with the previously discussed literature, we formulated four hypotheses, each aimed at assessing the research question from a different methodological viewpoint. The first hypothesis pertains to the participants’ performance, particularly referring to their score of correctly answered statements and their movement duration. This approach aligns with a body of research demonstrating the advantages of responding and initiating joint attention in HRI (e.g., Huang & Thomaz, 2011; Mehlmann et al., 2014). The second hypothesis investigates participants’ gaze-following behavior, referring to their classification decisions in correspondence with the robotic gaze cue. This view is grounded in work by Staudte and Crocker (2011), who found that participants trusted robotic gaze cues more than the factually correct spoken utterance, indicating a kind of automation bias. Based on that, our second assumption examines participants' strategic gaze-following behavior during both correct and incorrect robotic gaze hints, while interacting with the high-validity and low-validity robot. Pursuing this strategic path, the third hypothesis investigates participants' eye-tracking data, focusing on gaze patterns and eye movement predictability across the robotic conditions. While the first three approaches had a behavioral nature, the fourth hypothesis examines self-reporting responses, particularly about the social attributes of anthropomorphism, likability, intelligence, and trust. This fourth assumption, measured through a post-experiment questionnaire, provides a more subjective perspective on the participants' perception of the robots, as observed in various literature studies (e.g., Admoni & Scassellati, 2017; Mutlu et al., 2009). Accordingly, research was guided by the following hypothesis:

H1: *Participants will perform significantly better in interaction with the high-validity robot*

H2: *Participants' strategic bias to follow the high-validity robot leads to a kind of 'automation bias', causing users to follow its suggestion even if they are incorrect*

H3: *Participants will display more exploratory, unpredictable gaze behavior when interacting with the neutral robot, while the existence of referential gaze cues, albeit potentially incorrect, will lead to more predictable gaze patterns*

H4: *The reliability of a robot's gaze will positively influence the self-reporting social attributes of likability, intelligence, anthropomorphism and trust.*

Methodology

Design

The study used a 3 x 2 repeated-measures design. Three manipulated screen-based robots were used in interaction with a classification game that featured two levels of complexity. Accordingly, the first independent variable was robot identity, which implicitly varied in two key aspects: the presence of Initiating Joint Attention (IJA, or referential gaze) and the reliability of its gaze cues. In this case, IJA referred to the robot's expressions of eye movement to the classification categories, while reliability defined the degree to which these gaze behaviors were directed to the correct or incorrect classification category. The first robot, 'Ryan', directed referential gaze to the correct classification category for 80% of the trials (and 20% to the incorrect side). Throughout this paper, we refer to this robot by its name or primarily as the "high validity" robot. The second robot, 'Ivan', displayed referential gaze to the incorrect classification category in 80% of the statements (and 20% to the correct side). Thus, Ivan was defined as the "low validity" robot. The third robot, 'Carl', did not execute any IJA, serving as the 'neutral' robot. The second independent variable was statement difficulty, referring to the 'easy' and 'hard' statement categories. In total, 90 statements were presented across all conditions. The dependent variables encompassed four measurement groups: (1) performance metrics (e.g., accuracy score in classification game), (2) gaze-following data (e.g., whether participants followed the attentional cues), (3) eye-tracking data, capturing visual attention during interaction and (4) self-reporting measures (via a post-interaction questionnaire).

Participants

A total of 33 participants (15 male participants, 18 female participants, Age: $M = 23.00$, $SD = 2.44$) were recruited. Participants were selected using convenience sampling. All participants were students at the University of Twente. Fourteen participants lived in the Netherlands, and 19 participants lived in Germany. The inclusion criteria contained a sufficient level of English to understand the game statements and questionnaires, as well as normal or corrected-to-normal vision. No participants were excluded based on predefined criteria. The study and its procedures were approved by the local ethics committee of the University of Twente (request 250748).

Apparatus and Materials

Hardware

The technical setup for this experiment includes an HP Z1 computer with an AOC G2460PF 24-inch screen, which was connected to a Tobii Pro Fusion or Tobii X3 fixed eye-tracker. Additionally, the computer was connected to a Brio 4K streaming camera, which was mounted at the top of the screen. An iPad Air with a 9.7-inch screen was used to answer the questionnaire. Participants were seated at a desk in a monitored laboratory room. Screen height was individually adjusted so that its center aligned with the participant's eye level. Participants were approximately 50cm away from the monitor. A mouse and keyboard were connected to the computer.

Software

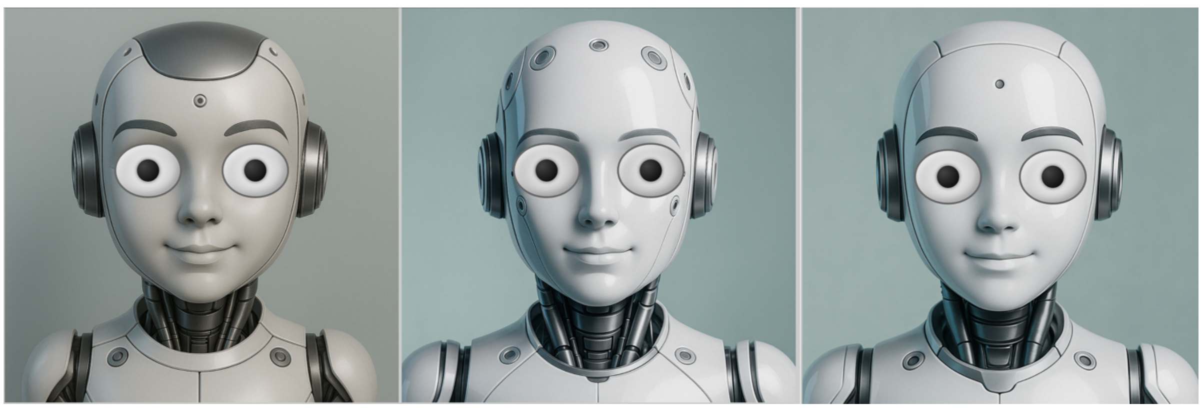
A questionnaire was designed and administered using Qualtrics Software (Qualtrics, 2025). The content of the questionnaire consisted of a briefing, informed consent, experimental information, and scale items to evaluate each robotic condition in the dimensions of anthropomorphism, likability, perceived intelligence, and trust. To measure the first three variables, three dimensions with five scale items of the Godspeed questionnaire were used. The three dimensions were selected as they represent core and validated metrics for assessing the key attributes of social perception in human-robot interaction (Bartneck et al., 2009). In addition, the choice of metrics aligns with frequently used constructs to evaluate social robots (Admoni & Scassellati, 2017). In the questionnaire, participants were for example asked to rate the robot between the scale items of 'fake' vs. 'natural' or 'incompetent vs. 'competent' (Bartneck et al., 2009). While the original Godspeed questionnaire encompasses five dimensions, the research team decided to exclude the dimensions of perceived safety and animacy as the robots displayed limited expressive abilities and only screen-based interaction. Lastly, the questionnaire used the brief 14-item version of Schaefer's Trust-Perception Scale for HRI (TPS-HRI). While the development of the full TPS-HRI involved the Army Research Laboratory, the 14-item concise version is frequently used in Human-Robot literature to measure how participants trust the robot (Schaefer, 2016). The TPS-HRI was answered on a seven-point Likert scale from 'Strongly Disagree' to 'Strongly Agree'. Items such as "The robot is reliable" or "The robot provides feedback" were included in this version. The research team used Tobii Pro Lab for screen recording, eye tracking calibration, and analysis of eye tracking variables. The experimental game and gaze control system were programmed and designed by the research team using HTML, CSS, JS, and Python as detailed below.

Robotic conditions

The three manipulated gaze conditions were allocated to similar-looking robot faces that display a high resemblance in their facial features, such as a slight smile, eyebrows, nose, robotic ears, and a slightly positive facial expression. The robots were artificially generated using OpenAI's DALL·E 2 image-generation model via GPT-4 (OpenAI, 2025). They were specifically designed to feature characteristics that already correspond to real humanoid robots, such as the iCub (Metta et al., 2010). Prompting statements to generate robotic pictures can be found in Appendix 1. Figure 2 shows an image of each generated robot.

Figure 2.

The Three Robots used in the Experiment overlaid with the Eyes of the Interactive Gaze System.



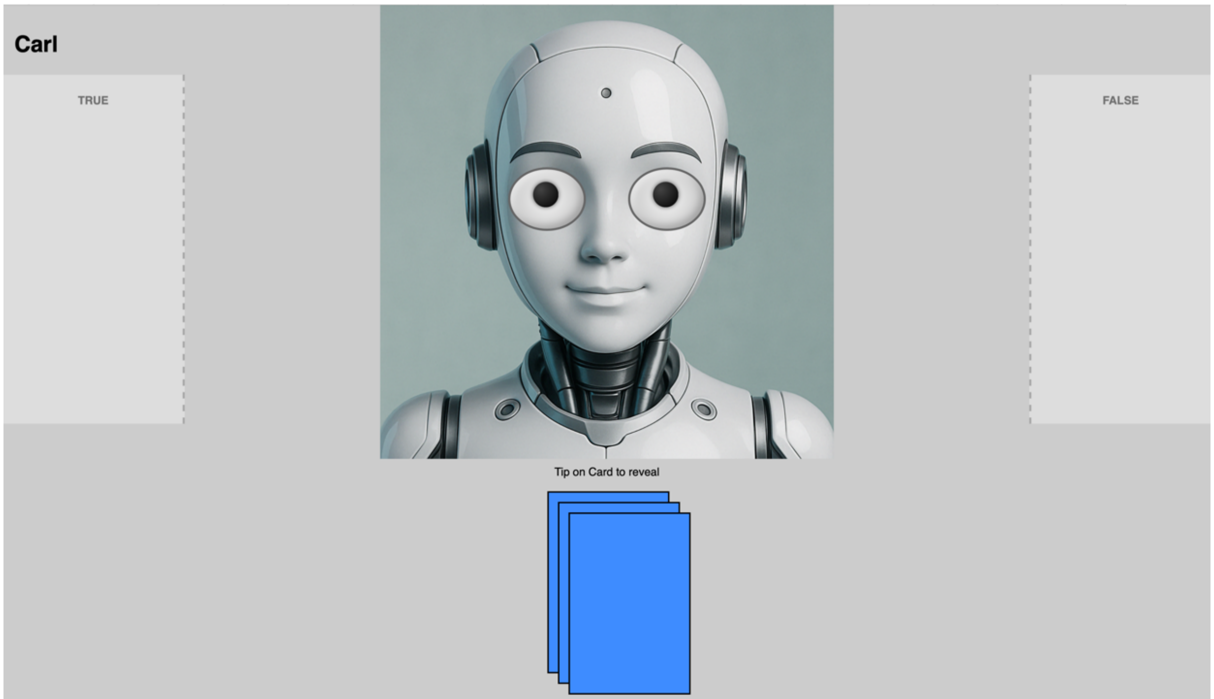
Note. Each static robotic picture was generated using OpenAI's DALL-E 2 Image Generation Model. Ryan (high validity) on the left side, Ivan (low validity) in the middle, and Carl (neutral/ control) on the right side.

The experimental game

The experiment involved playing a classification game with a screen-based robot. A picture of the experimental interface is shown in Figure 3. Participants were able to reveal a card by clicking on the blue stack of cards and move and drop the card to one of the two categories on the left or right side based on their intuition whether the statement was true or false. The mouse was used to move a card around. The stack of cards consisted of 90 statements, with 45 categorized as easy and 45 categorized as hard. The statements were chosen from a public database of general facts hosted by the machine learning platform Hugging Face (L1Fthrasir, 2023). The list of all statements can be found in Appendix 5. An example of a simple statement is “The Sun is more massive than the Earth”. A more complex question was “The respiratory system prevents the exchange of gases between the body and the environment”. The researcher initially selected and categorized each statement. Subsequently, a third-party reviewer independently assessed and validated the categorization to ensure reliability. The order of statements was randomized entirely for each participant. However, the algorithm considered each robot to receive the same number of easy and hard questions. To enable bidirectional real-time communication between user events, such as a card reveal or card drop, the program used a WebSocket API on the local connected network, allowing the gaze control system and classification game to interact in real-time.

Figure 3.

The Interface of the Experimental Game and real-time animated Eyes (Gaze Control System) that shows the current Robot, the Robot's Name, the Cards to reveal, and the Categories True (left side) and False (right side).



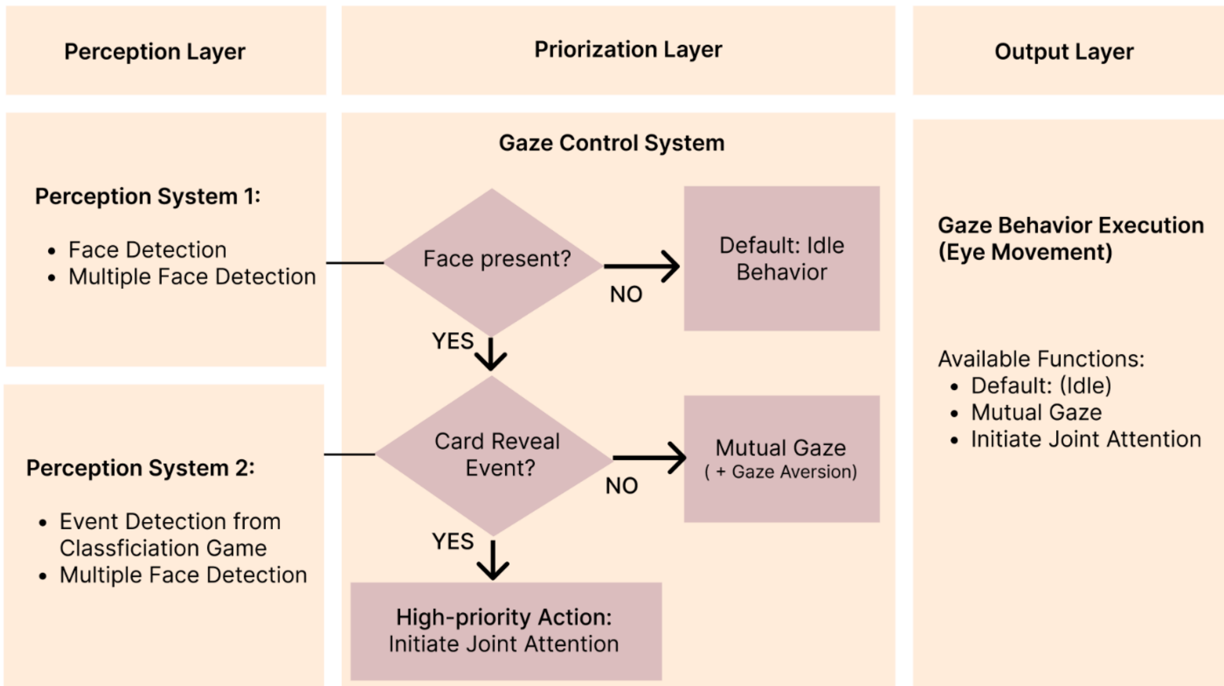
The gaze control system

The gaze control system was built and implemented using HTML, CSS, JavaScript, and Python. Using a static robotic picture, the interactive system encompasses the eyes and pupils. As shown in Figure 4, the input system, written in Python and JavaScript, was locally connected using the WebSocket API. The program received data input from the Brio 4K camera to detect face and head position in real-time. Additionally, it received user events such as game start, card reveal, or card drop. Such event messages not only contained the event name, but also additional calculated information such as the statement, the correct side, or the following robotic condition.

All these input parameters were checked and validated in the gaze control system, whose output determined the robot's gaze behavior in real-time. During the experiment, the gaze system (output) was limited to three gaze behaviors: mutual gaze, gaze aversion, and Initiating Joint Attention (IJA). Mutual gaze describes a condition in which the pupils of the robot are positioned in alignment with the coordinates of the face in front of the screen. This created an illusion in which it looked as if the robot was looking at the participant. In this situation, the robotic eyes followed the participant's head movement without a recognizable latency. Avoiding staring behavior, the robot also applied gaze aversion within mutual gaze at randomized time intervals between 1000 and 3000 milliseconds. Gaze aversion can be described as a periodically brief, fixed-duration gaze shift toward a randomly determined off-center point. Gaze Aversion was biased towards vertical rather than horizontal displacement to avoid confusion with IJA behavior. Finally, IJA describes a triggered gaze behavior, in which the robot smoothly shifts its gaze from its current position towards a designated direction and maintains its gaze fixed on that side for 2000 milliseconds unless it smoothly transitions its gaze back towards the user's currently detected face position.

Figure 4.

Overview of the Gaze Control System with Its Three Layers.



Note. The Perception layers detected real-time information from the participant's face and the experimental game. Information was processed and prioritized in the Priorization Layer to finally calculate the eye movement, which was applied from the output layer.

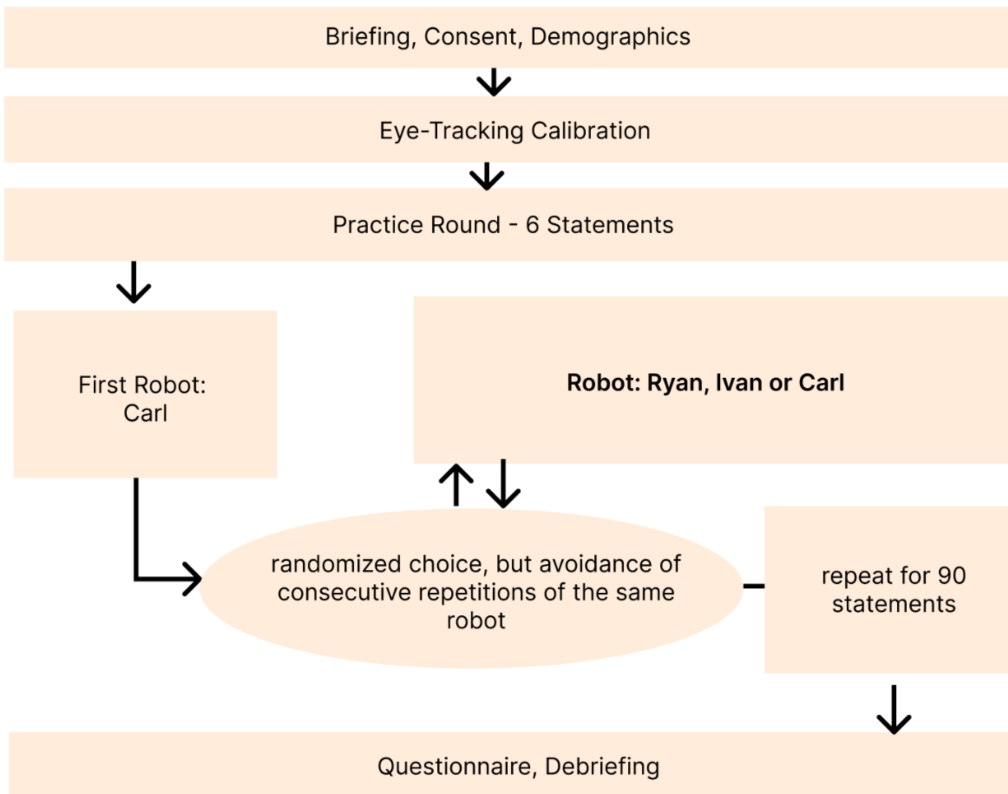
During the game, mutual gaze was the default active behavior when a user was present, and no higher-priority actions occurred. IJA was triggered for a card reveal event and, after a short delay, explicitly interrupted and overridden mutual gaze until the IJA action was complete. IJA was executed in the high-validity (Ryan) and low-validity (Ivan) robots only. This is also visualized in Figure 4, which shows the algorithmic prioritization.

Procedure

Each participant was recruited individually and invited to sit in front of the researcher's laptop at a desk. After the introduction, the participant received the iPad with the Qualtrics Questionnaire. After provision of electronic consent and a study briefing in the questionnaire, participants received a second oral briefing about the experimental procedure as well as the opportunity to ask remaining questions before the experimental game. If there were no more questions, participants were instructed to begin calibrating the eye tracker by following a dot on the screen. After successful calibration, defined by an average calibration accuracy of less than 0.5 degrees of visual angle, participants saw the web interface and were able to enter their participant ID. Before the actual game, each participant was instructed to participate in a practice round, in which they had to classify six statements. Participants were not able to see one of the robots during the practice session. With the end of that session, participants were able to start the real game. After the practice session, the researcher left the monitored room so that the participant could play the game undisturbed. As visualized in Figure 5, participants always started with the neutral robot (Carl), which was then randomly switched after each card drop, considering that no robot appears twice in a row. This randomized process was repeated for a total of 90 statements. After completing the statements, a pop-up window informed participants about the end of the experimental game. Thereupon, participants were required to complete a post-questionnaire with dimensional questions for each robot. The questionnaire order began with the neutral robot (Carl), continued with the high-validity robot (Ryan), and ended with the low-validity robot (Ivan). A debriefing followed.

Figure 5.

The procedure of the entire experiment, starting with the informed consent and ending with an evaluation questionnaire and debriefing.



Data Analysis

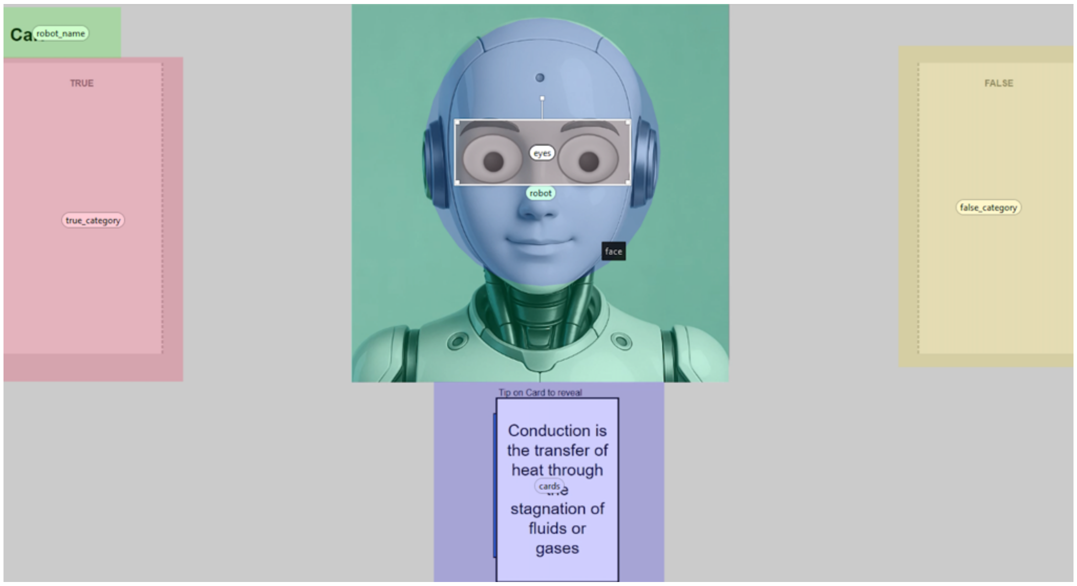
All data analysis was conducted using Python in Visual Studio Code and R in RStudio. During the experiment, a CSV file was created for each participant. This file contains event and performance-related information as well as specific timestamps for each piece of information. A Python script was used to combine each participant's file into an overall CSV file.

An additional Python script was used to transform the raw Qualtrics data files into a suitable CSV file, which included demographic variables, consent information, and numerically converted values. In this script, three reversely coded trust items (Item 9, Item 11, Item 14) from the TPS-HRI scale (Schaefer, 2016) were converted appropriately.

For the eye-tracking data, specific areas of interest were marked on the screen recordings in Tobii Pro lab. Figure 6 shows a picture of the relevant areas of interest. For every participant, a TSV file was downloaded separately from Tobii Pro lab. A Python script was used to combine the relevant eye-tracking data for each participant with the event CSV file. This script, located in Appendix 7, enables differentiation of eye-tracking data for each game event.

Figure 6.

A Screenshot of the Screen recording with the Areas of Interest highlighted in different Colors.



Note. The areas ‘true_category’ and ‘false_category’ on the left and right sides were combined into ‘classification_category’. The ‘robot’ AOI encompasses the green shape, the ‘face’ AOI encompasses the oval ‘slate-blue’ shape, and the ‘eyes’ AOI shapes the central, white-bordered rectangle.

Performance and Move Duration Analyses

The analysis of performance data encompassed the proportion of correctly answered statements (accuracy score, ranging from 0 to 1) and move duration, which defined the time (seconds) it took participants to drop a card into a classification category after card reveal. The effects of these variables were examined using a 3 (Robot: High-validity, low-validity, neutral) x 2 (Difficulty: Easy, hard) repeated-measures Analysis of Variance (ANOVA). Significant main effects were further investigated using post-hoc comparisons with Bonferroni correction.

Gaze following analysis

For the gaze following analysis, a participant's gaze follow was defined for every trial, where the participant moved the item to the identical side to which the robot applied referential gaze before. Since the neutral robot did not use any referential gaze, this was only calculated for the high-validity and low-validity robots. Descriptive statistics (frequencies and percentages) were computed for participants' gaze-following behavior (followed, not followed). Participants' data were segmented by robotic identity, gaze correctness (I.e., did the robot look to the correct side?), and difficulty level (hard or easy statements). Gaze following behavior was further analyzed using the framework of signal detection theory (SDT). Therefore, the two key metrics of sensitivity (*d'*) and response criterion (*c'*) were computed. To investigate how these measures were affected by robotic identity and difficulty level, a 2 (Robot: High-validity, low-validity) x 2 (Difficulty: Easy, hard) repeated-measures ANOVA was calculated for the metric *d'* and *c',* respectively.

Eye-Tracking analyses

Participants' visual attention and gaze patterns were analyzed using a variety of eye-tracking metrics. Heatmaps were created to visualize participants' gaze concentrations during each robotic and difficulty condition. Previously defined AOIs (see Figure 6) were used to investigate total dwell time and frequency of visits. The influence of the experimental conditions on these metrics was assessed using a series of 3 (Robot: High-validity, low-validity, neutral) x 2 (Difficulty: Easy, hard) repeated-measures ANOVAs. Subsequently, an advanced AOI transition analysis was applied to examine participants' attentional shifts and gaze transitions. This was further visualized for the AOIs using transition probabilities. Chi-squared tests were employed to determine if the observed transition patterns differed significantly as a function of robot identity and difficulty. Finally, a Recurrence Quantification Analysis (RQA) was applied to assess the predictability and structure of participants' gaze patterns. The RQA computed a Determinism (DET) score, which quantifies the extent to which a pattern or sequence of states repeats itself (Anderson et al., 2013). A high DET score, usually for values above 70%, indicates a structured and repeated gaze frequency. A lower DET score, usually less than 40%, suggests more random, unstructured gaze movement. A 3 (Robot: High-validity, low-validity, neutral) x 2 (Difficulty: Easy, hard) repeated-measures ANOVA was conducted on the DET scores.

Self-report questionnaire analysis plan

For the self-report analysis of Qualtrics data, ratings for the following variables were computed respectively for each robot: Anthropomorphism, Likability, Intelligence, and Trust. To compare participants' subjective ratings across the three robots (High-validity, low-validity, neutral), a series of four separate one-way repeated-measures ANOVAs was conducted. Significant effects were further examined using post-hoc pairwise comparisons with a Bonferroni correction.

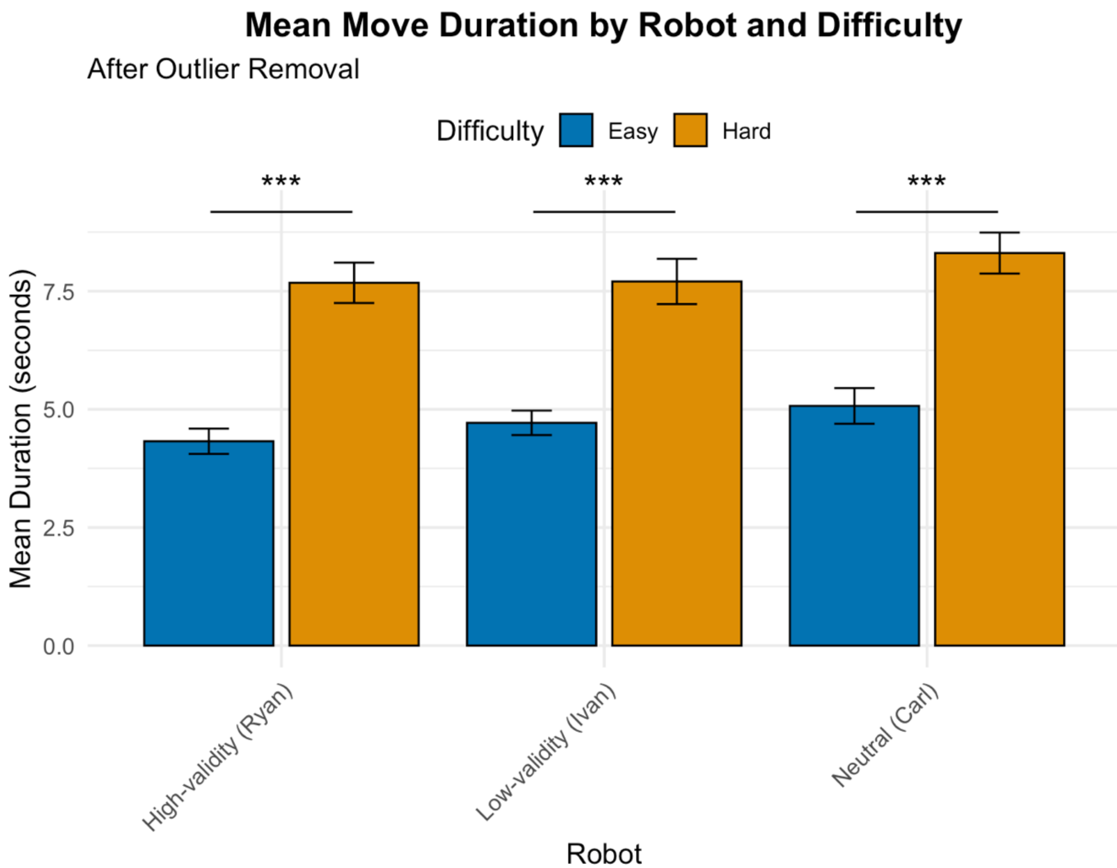
Results

Performance Analysis

The performance data encompasses the participants' accuracy scores of correctly answered statements as well as their move durations to classify a statement. Based on a 2.5 standard deviation rule applied to each participant's data, 82 trials (2.76% of the total) were removed as outliers from the move duration variable prior to the main analysis. Participants scored consistently higher on 'easy' (M = 0.93, SD = 0.25) compared to 'hard' (M = 0.62, SD = 0.47) statements across all robots. As visualized in Figure 7, this trend is also reflected in the movement duration scores, with participants showing shorter durations for statements categorized as easy (M = 4.82s, SD = 3.94) compared to those marked as hard (M = 8.55s, SD = 5.92s).

Figure 7.

Bar Chart of Mean Move Duration across the Three Robots and Easy and Hard Difficulty Levels.



Note. The bar chart displays the mean time in seconds that participants took to complete a trial for each of the three robots and two difficulty levels. The data shown are from trials remaining after the removal of outliers. Error bars represent ±1 standard error of the mean. *** $p < .001$.

A 3 (Robot: High-validity, low-validity, neutral) x 2 (Difficulty: easy, hard) repeated-measures ANOVA was conducted to examine effects on task accuracy percentage. The analysis revealed no significant main effect of robot on the accuracy score ($F(2, 64) = 0.26$, $p = .76$, $\eta g^2 = .003$), suggesting that the performance did not differ significantly across the three robots. As expected, there was a significant main effect of Difficulty ($F(1, 32) = 333.66$, $p < .001$, $\eta g^2 =$ .670), with participants displaying significantly higher accuracy on easy tasks (M = 93.2%, SD = 7.25%) compared to 'hard' tasks (M = 62.5%, SD = 13.8%; Mean Diff = 30.8, 95% CI [27.3, 34.2]). The Robot x Difficulty interaction effect was not statistically significant ($F(2, 64) = 2.75$, $p = .072$, generalized $\eta^2 = .023$).

For move duration, a 3 (Robot: High-validity, low-validity, neutral) x 2 (Difficulty: easy, hard) repeated-measures ANOVA was conducted. The analysis revealed a statistically significant main effect of robot on move duration ($F(2, 64) = 6.19$, $p = .003$, $\eta g^2 = .017$). Post-hoc pairwise comparisons with Bonferroni correction displayed participants having significantly longer move durations when interacting with the neutral robot (M = 6.69s, SD = 2.83s) compared to the high-validity robot (M = 6.00s, SD = 2.64s; Mean Diff = 0.69s, 95% CI [0.15, 1.22], $p = .008$). No other pairwise differences for the main effect reached statistical significance (High-validity vs. low-validity: $p = .712$; low-validity vs. neutral: $p = .097$). As expected, there was also a significant main effect of Difficulty, $F(1, 32) = 234.96$, $p < .001$, $\eta g^2 = .352$. Participants exhibited significantly shorter move durations on 'easy' tasks (M = 4.70s, SD = 1.77s) compared to 'hard' tasks (M = 7.90s, SD = 2.56s; Mean Diff = -3.19s, 95% CI [-3.62, -2.77]). The Robot x Difficulty interaction effect was not statistically significant ($F(2, 64) = 0.79$, $p = .459$, $\eta g^2 =$ .001).

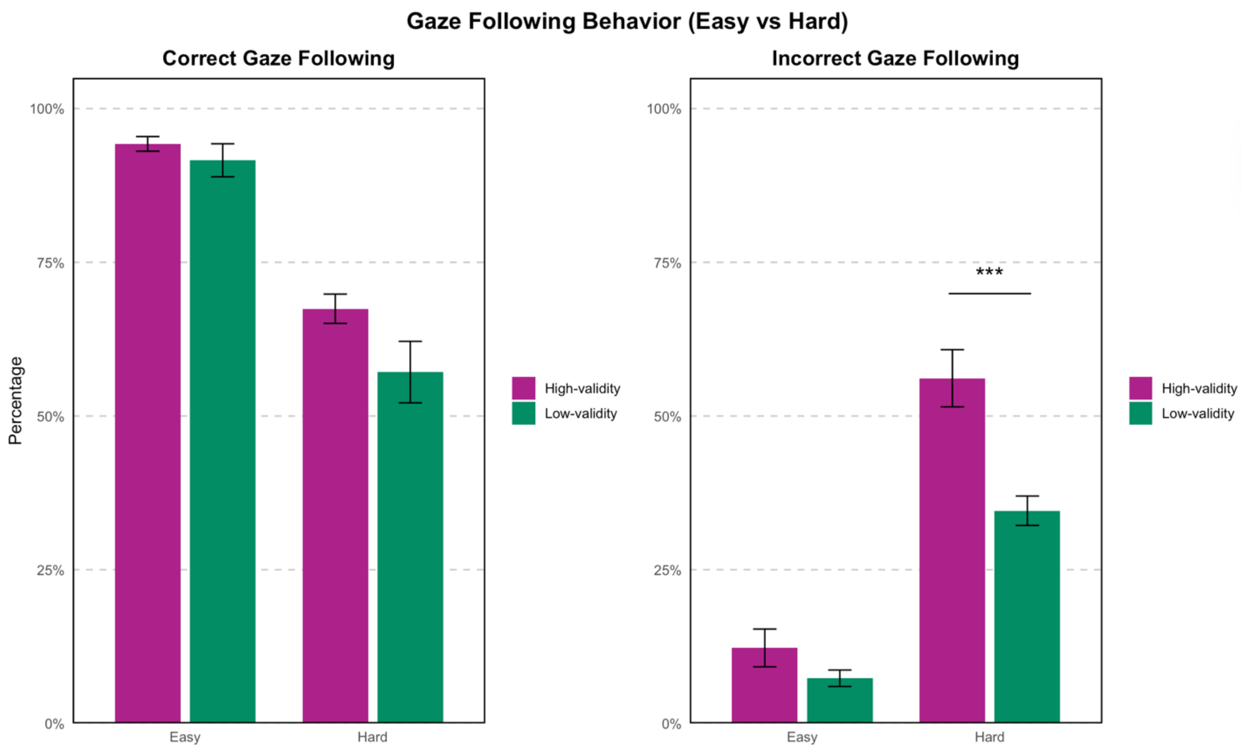
Gaze Following Analysis

The experimental design for the high-validity robot (Ryan) aimed for its gaze cues to point in the correct direction in 80% of instances, with the remaining 20% being incorrect. For the low-validity robot (Ivan), the gaze was programmed to look in the incorrect direction in 80% of cases. To enhance internal validity, the assignment process of correct and incorrect gaze hints was randomized by chance for each trial. Hence, analysis of the collected data revealed that gaze cues from Ryan were correct in 77.2% of trials (n = 770 out of 998), while Ivan displayed gaze towards the correct side in only 20.9% of trials (n = 205 out of 982). A 'gaze follow behavior' was defined as an instance where a participant's choice of side was congruent with the direction of the robot's referential gaze cue.

Figure 8 illustrates the proportional frequency with which participants chose the classification category indicated by the robotic gaze cue. When the high-validity robots' gaze was correct, participants followed in 80.8% of cases (n = 622 out of 770 total correct gaze trials; easy statements: 94.3%, hard statements: 67.4%). In terms of incorrect robotic execution from the high-validity robot, participants followed its misleading gaze in 34.2% of cases (n = 78 out of 228 total incorrect gaze trials; easy statements: 12.3%, hard statements: 56.1%). For the low-validity robot, when gaze was correct, participants followed in 75.1% of cases (n = 154 out of 205 total correct gaze trials; easy statements: 91.6%, hard statements: 57.1%). When gaze was incorrect, participants followed this misleading gaze in 21.2% of cases (n = 165 out of 777 total incorrect gaze trials; easy statements: 7.3%, hard statements: 34.6%).

Figure 8.

Bar Charts showing the Percentage of Participants that followed the Robotic Gaze Hints in their Decision-Making Process for the High-Validity (Ryan) and Low-Validity (Ivan) Robots across Easy and Hard Statements.



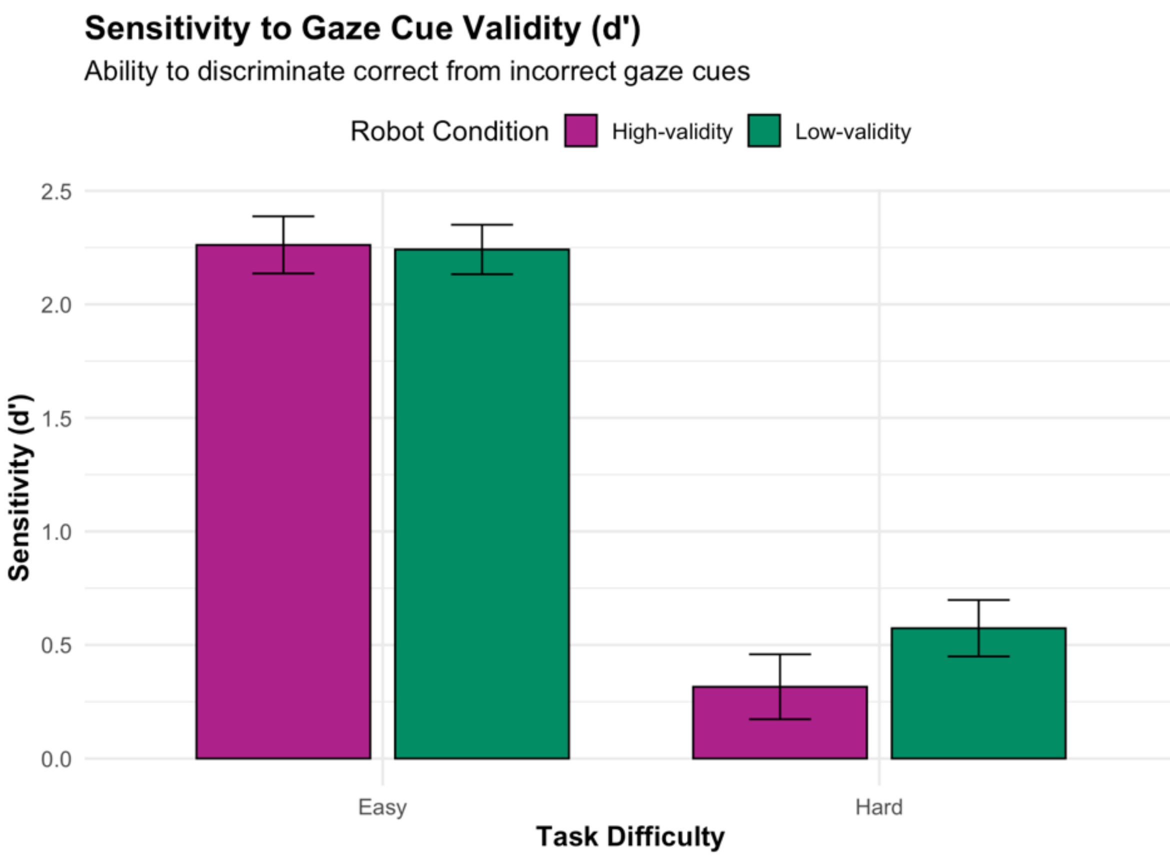
Note. The bar chart on the left side visualizes this behavior, when the robotic gaze hints pointed to the correct category, while the right chart indicates behavior when robotic cues were misleading. Error bars represent ±1 standard error of the mean (SEM). *** $p < .001$.

To gain detailed insight into participants' gaze-following behavior, the inferential analysis was approached from the perspective of Signal Detection Theory (SDT). This method allowed the separation of two key processes. The metric of sensitivity (*d'*) represents how well participants could tell whether a robot's gaze cue pointed to the correct location. Secondly, the response criterion (*c*') measures the participants' general bias to follow the gaze cue. For *d*', a value of zero would indicate a complete inability to distinguish between correct and incorrect gaze cues. Hence, a value of 1.0 is considered to reflect a moderate sensitivity, while a value of 3.0 or higher would indicate a near-perfect discrimination. For the *c'* value, the number zero would represent a neutral bias or a neutral strategy of the participants. In alignment with SDT, every *c'* value below the zero line can be defined as liberal bias, a tendency to follow the robot. Positive values instead can be defined as conservative bias, a tendency not to follow the robotic gaze cue. Although this is not strictly bounded, values for *c'* typically fall between -1 and +1, with values further from zero indicating a stronger bias. For both sensitivity and response criterion, two separate 2 (Robot: High-validity vs. low-validity) x 2 (Difficulty: Easy vs Hard) repeated-measures ANOVAs were computed.

The analysis of sensitivity revealed a significant main effect of difficulty ($F(1,32) = 137.64$, $p < .001$, $\eta g^2 = .54$) with participants being significantly less able to distinguish between correct and incorrect gaze cues during the 'hard' category ($M = 0.45$, $SD = 0.77$) compared to the 'easy' condition (M = 2.25, SD = 0.67; Mean Diff = 1.81, 95% CI [1.49, 2.12]). However, no significant differences emerged between the robot ($p = .320$) and sensitivity, nor was there a significant interaction between the robot and difficulty ($p = .169$). This is highlighted in Figure 9, where participants achieved a significantly higher sensitivity across the easy condition compared to the hard one. These findings suggest that the robot did not affect how well participants could distinguish between correct and incorrect gaze cues. Instead, task difficulty was the primary driver impairing participants' ability to evaluate the quality of gaze hints.

Figure 9.

Bar Chart of the Mean Sensitivity (d') from Signal Detection Theory Method as a Function of Task difficulty for the High-validity and Low-validity Robot.



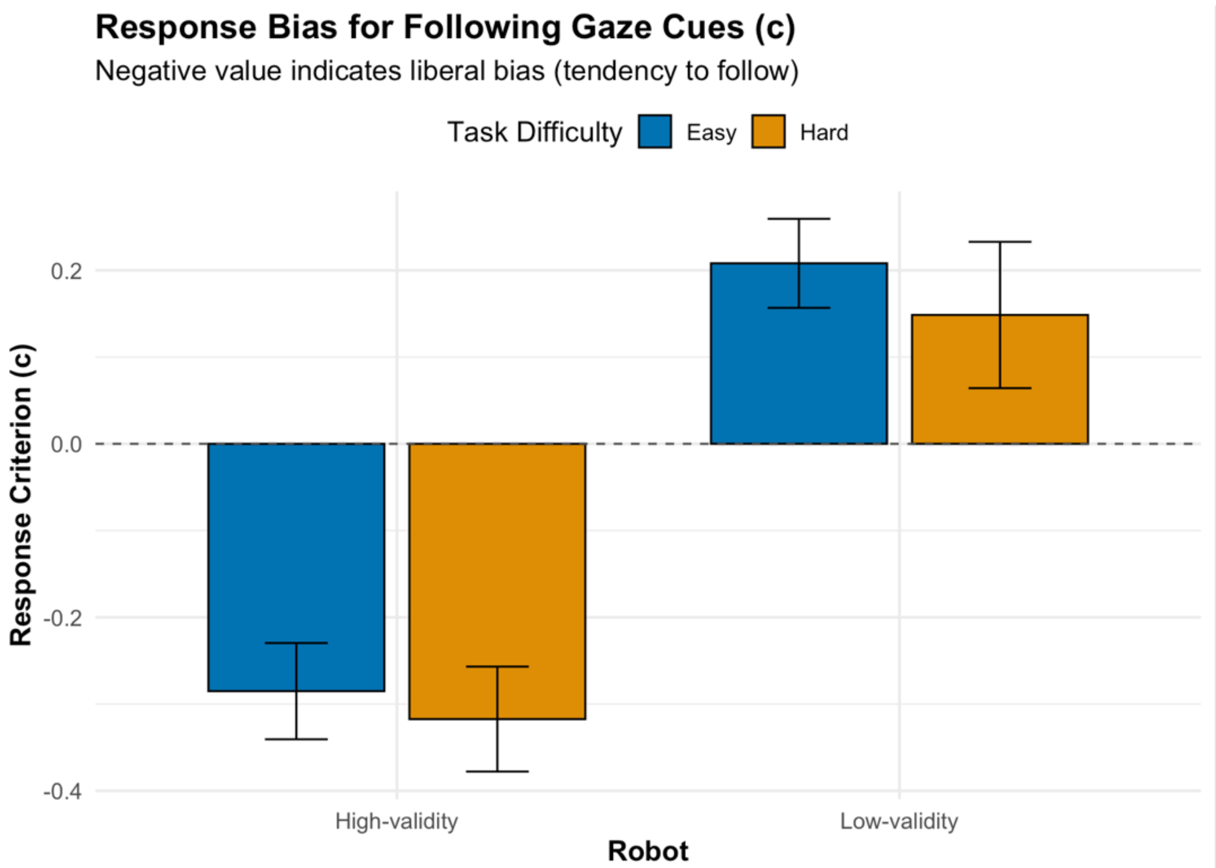
Note. Bars represent mean sensitivity scores for participants while discriminating between correct and incorrect gaze cues. Higher values indicate better discrimination. Error bars represent ±1 standard error of the mean (SEM).

However, the analysis of response criterion showed a significant main effect of robot ($F(1,32) = 37.67$, $p < .001$, $\eta g^2 = .28$). Consequently, participants adopted a different strategic bias towards the robots. As visualized in Figure 10, participants were significantly more inclined to follow gaze cues from the high-validity robot compared to the low-validity robot (Mean Diff =

0.48, 95% CI [0.32, 0.64], $p < .001$). Both bars of Figure 10 for the low-validity robot are positive, showing participants' conservative bias towards the unreliable gaze ($M = 0.18$, $SD = 0.40$). For the high-validity robot, the two bars position themselves consistently below the zero line, reflecting participants' willingness to follow its gaze ($M = -0.30$, $SD = 0.33$). This finding suggests that participants developed a consistent strategic preference for trusting the reliable, high-validity robot. Apart from that, neither task difficulty ($p = .421$) nor the interaction between robot and difficulty ($p = .814$) had a significant effect, reflecting that this strategic preference was adopted consistently regardless of task difficulty.

Figure 10.

Bar Chart of Response Bias (c') criterion from Signal Detection Theory Method as a Function of Task difficulty for the High-validity and Low-validity Robot.



Note. Bars represent participants' average response bias. Negative values reflect tendencies to follow the cue, while positive values reflect a tendency to ignore the cue. Error bars represent ±1 standard error of the mean (SEM).

To summarize, the above analyses and use of Signal Detection Theory revealed that task difficulty significantly reduced participants' ability to distinguish between correct and incorrect cues (sensitivity), regardless of the robot identity. However, the strategic response bias showed strong preferences to follow the high-validity robot's gaze more than that of its counterpart (low-validity), regardless of task difficulty, which indicates a strategic trust in the more reliable robot, even when cues were misleading.

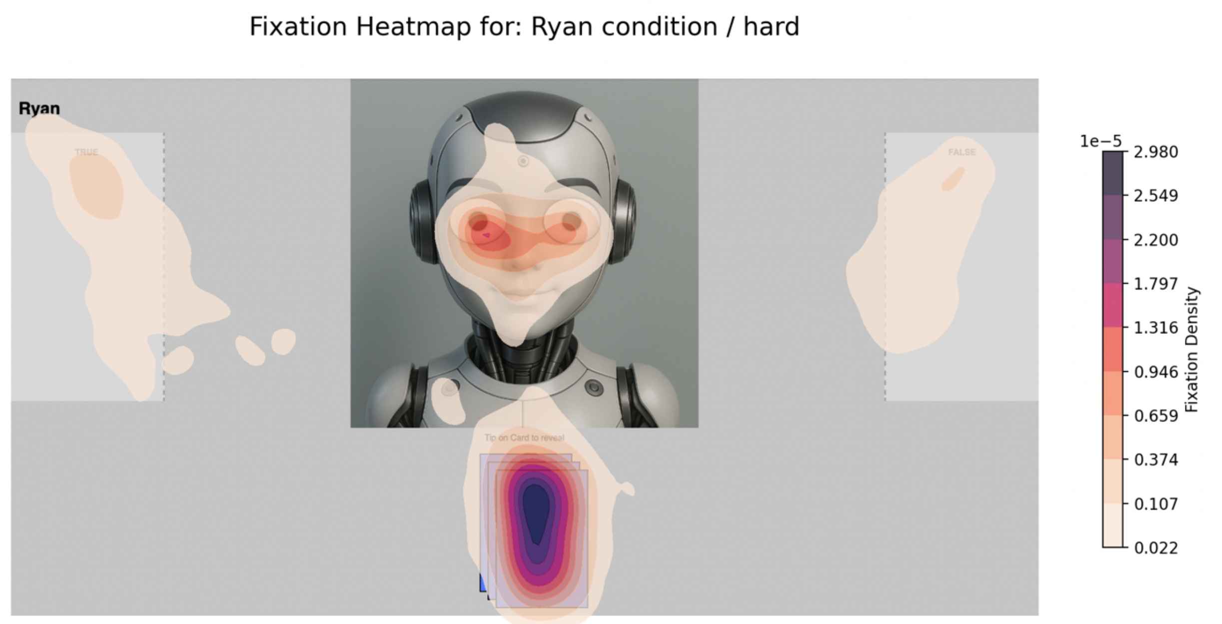
Eye-tracking analysis

Analysis of Areas of Interest

To examine participants' visual attention, gaze data were analyzed based on predefined Areas of Interest (AOI). As mentioned in the methodology, the main AOI groups encompassed the blue cards ("cards"), the two classification category fields ("Classification"), and the robot (represented by the AOIs of "eyes", "face" and "robot", while the last one accounted for the entire robot picture with the grey background). An overview of the AOI can be found in the data analysis part of the methodology (Figure 6). To illustrate visual attention, Figure 11 shows a heatmap of where participants looked most frequently. Heatmaps for further conditions can be found in Appendix 4. Figure 11 shows the high-validity robot for 'hard' statements only but can be considered representative as basic visual allocation across all conditions.

Figure 11.

Heatmap of Participants' Visual Fixation for the High-Validity Robot During Hard Statements.



Note. This heatmap shows an overview of participants' visual allocation. Higher concentration of fixations is represented by 'darker' colors.

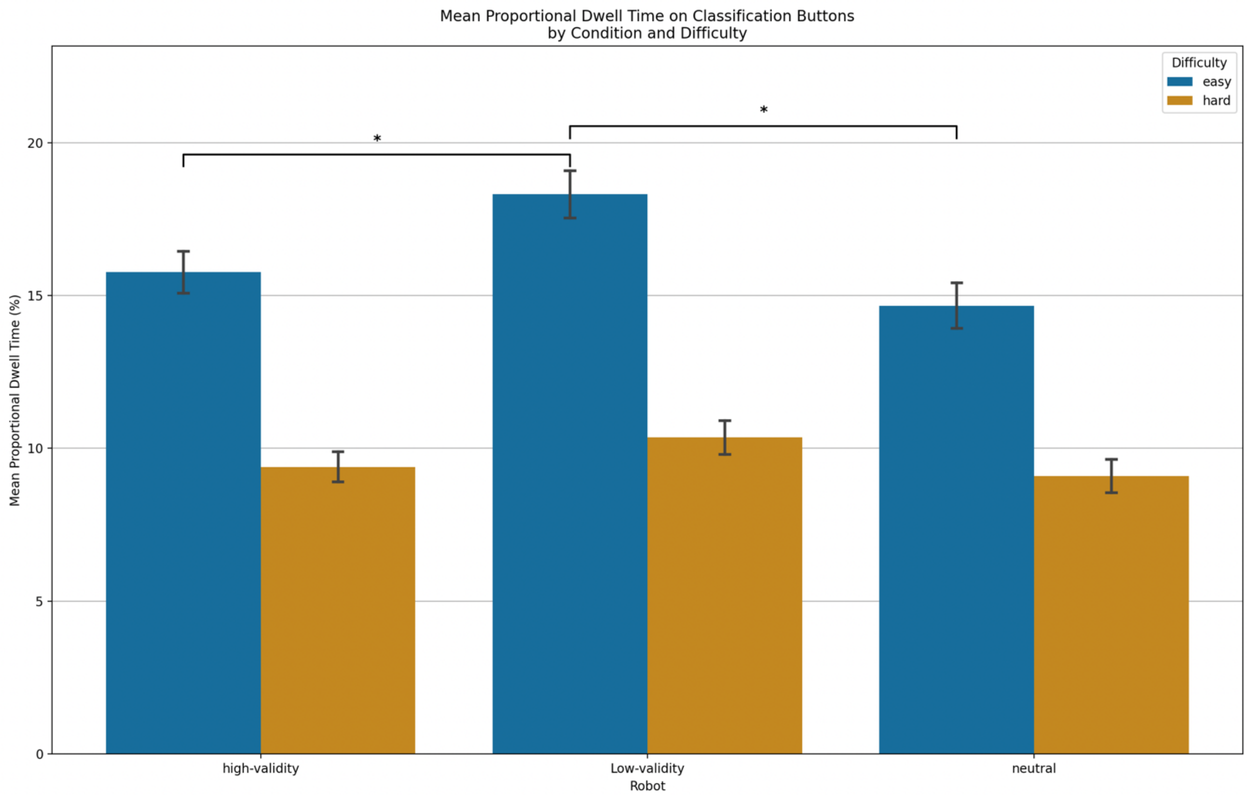
To investigate participants' eye movements in alignment with the defined AOIs, we started the eye-tracking data analysis with an examination of proportional dwell times (the percentage of total trial time spent looking at an area) and fixation frequency (the percentage of total fixations within an area). As highlighted in Figure 11, our analysis focused on three representational areas: the cards, the robotic face, and the classification category (including the right and left classification sides). For each metric and AOI, we conducted a 3 (robot) x 2 (difficulty) repeated-measures ANOVA.

Starting with the cards, a dwell analysis revealed only a significant effect of difficulty ($F(1, 31) = 200.27$, $p < .001$, $\eta g^2 = .144$), indicating that participants spent a larger amount of time focusing on hard statements (M = 58.7%, SD = 10.9%) compared to easy ones (M = 44.1%, SD = 10.1%; Mean Diff = -14.6, 95% CI [-17.0, -12.2]). No significant differences between the robots (p = .087), nor an interaction effect (p = .119) were found. This trend continues for the proportional fixation frequencies, showing only a significant impact of difficulty level ($F(1, 31) = 230.27$, $p < .001$, $\eta g^2 = .196$) but no effect of robot (p = .231), or an interaction (p = .161)

An analysis of participants dwell time towards the classification categories indicated a significant effect of difficulty ($F(1, 31) = 78.51$, $p < .001$, $\eta g^2 = .172$) as well as a substantial impact of robot identity ($F(2, 62) = 5.71$, $p = .005$, $\eta g^2 = .018$). However, the interaction was not significant (p = .280). Post-hoc comparisons showed that participants spent significantly more time examining the classification categories when interacting with the low-validity robot (M = 15.1%, SD = 5.4%) compared to both the high-validity robot (M = 12.5%, SD = 4.3%; Mean Diff = -2.6, 95% CI [-3.0, -1.0], p = .012), and compared the neutral robot (M = 12.0%, SD = 4.9%; Mean Diff = 3.1, 95% CI [1.0, 4.0], p = .013). This finding, also highlighted in Figure 12, indicates participants' uncertainty and their stronger verification after misleading gaze cues from the robot. A similar result was obtained in the proportional fixation frequency analysis, showing a significant effect of robot ($F(2, 62) = 5.55$, $p = .006$, $\eta g^2 = .004$) and difficulty ($F(1, 31) = 94.97$, $p < .001$, $\eta g^2 = .080$) with no interaction effect (p = .543). For the significant effect of difficulty, participants dwelled longer during hard statements (M = 16.2%, SD = 5.6%) than easy statements (M = 10.2%, SD = 3.5%; Mean Diff = 6.0, 95% CI [5.0,8.0], p < .001).

Figure 12

Bar Chart of the Proportional Dwell Time that Participants spent on the AOI Classification Categories.

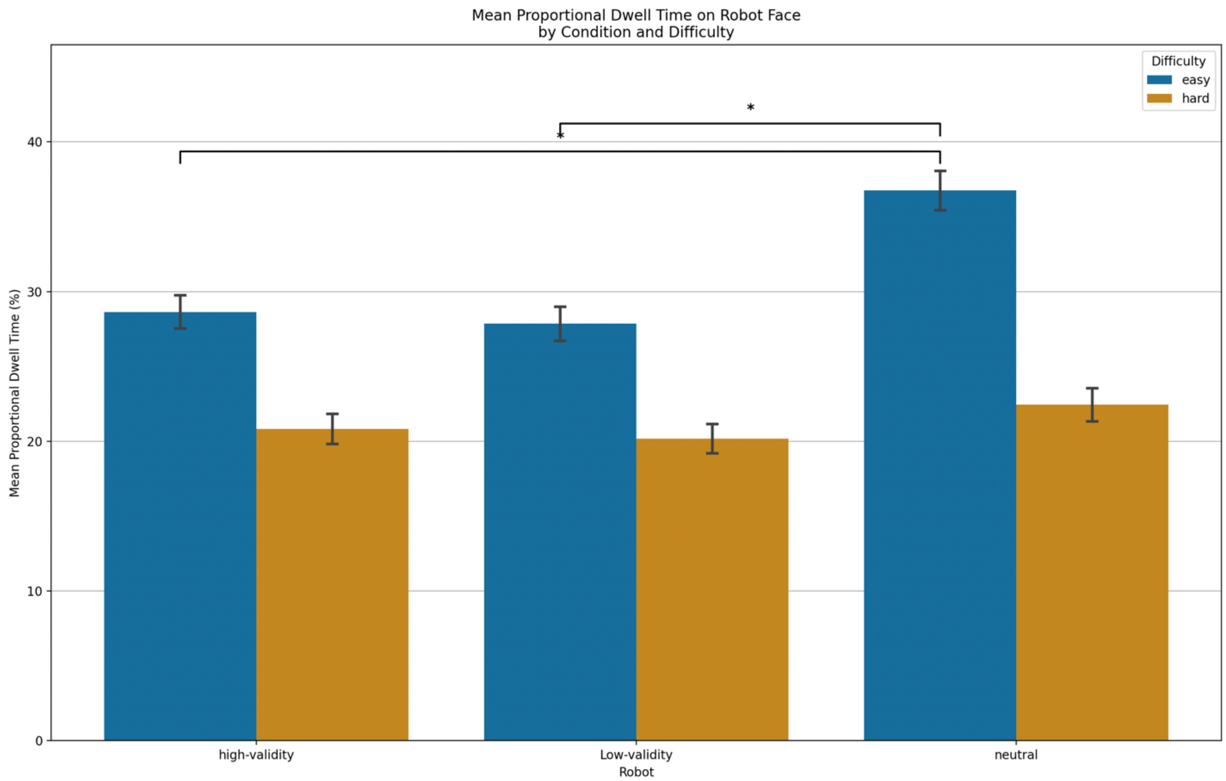


Note. *The classification category AOI includes the left and right classification boxes on the sides of the experimental game.* Outliers greater than 2.5 standard deviations from the mean were removed prior to analysis. Error bars represent ±1 standard error of the mean (SEM). *$p < .05$.

Finally, the analysis of dwell time for the "face" AOI showed a significant main effect of both robot identity ($F(2, 62) = 10.84$, $p < .001$, $\eta g^2 = .019$) and difficulty level ($F(1, 31) = 82.08$, $p < .001$, $\eta g^2 = .081$). In contrast to the previous analyses, an interaction effect between robot and difficulty level was also found ($F(2, 62) = 11.49$, $p < .001$, $\eta g^2 = .010$), suggesting that the amount of time participants looked at the robot was influenced by statement complexity. Post-hoc testing revealed that participants spent a significantly greater proportion of time looking at the face while interacting with the neutral robot (M = 23.3%, SD = 9.8%) compared to both the high-validity robot (M = 18.8%, SD = 8.1%; Mean Diff = -4.5, 95% CI [-7.0, -2.0], p = .007) and the low-validity robot (M = 18.2%, SD = 8.7%; Mean Diff = 5.1, 95% CI [3.0, 9.0], p = .003). This is also visualized in Figure 13. Further, this finding found support from the analysis of proportional fixation frequency, showing a significant effect of robot ($F(2, 62) = 5.03$, $p = .009$, $\eta g^2 = .009$) and difficulty ($F(1, 31) = 84.09$, $p < .001$, $\eta g^2 = .126$) as well as an interaction effect ($F(2, 62) = 4.46$, $p = .015$, $\eta g^2 = .004$). Post-hoc analysis of main effects revealed that while participants fixated more on the face during hard statements for all robots, this effect was most pronounced for the neutral robot.

Figure 13.

Bar Chart of the proportional Dwell Time that Participants spent on the Face AOI during the experiment for the three robots for easy and hard statements.



Note. Outliers greater than 2.5 standard deviations from the mean were removed prior to analysis. Error bars represent ±1 standard error of the mean (SEM). **p* < .05.

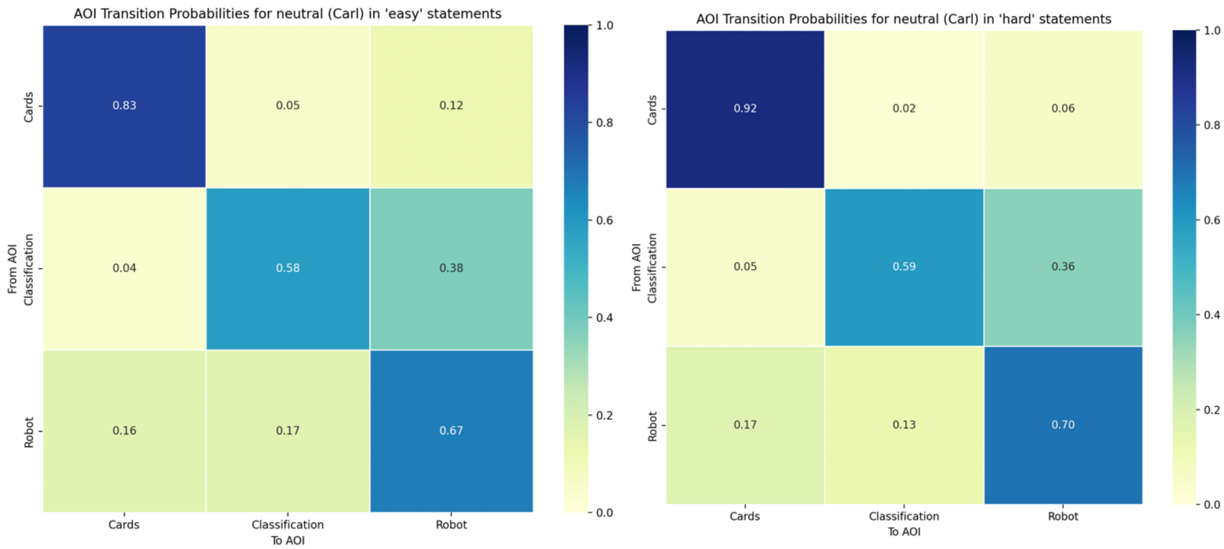
Advanced AOI Transition Analysis

The previous and initial eye-tracking analysis focused on where and how frequently participants looked at the AOIs. Taking it a step further, this section applied an advanced AOI transition analysis to investigate the dynamics of attentional shifts, particularly the transitions between AOIs and their directionality. Hence, the previously examined AOIs - "cards", "face", and "classification categories" - were used. Figure 14 illustrates two representative transition probability maps for the neutral robot. Additional transition maps of the high-validity and low-validity robots can be found in Appendix 4 (Figure 2).

Accounting for an investigation of fixation-sustaining and strategical transition patterns, two underlying behavioral gaze patterns were identified using the transition maps. These strategic patterns consistently dominated across all complexity levels and robotic identities. First and most frequently, a dominant self-transition to stay at the 'cards' AOI was observed. As visualized in Figure 14, the cards AOI became a gaze focus, with participants being extremely likely to remain at that area with their gaze, even more likely for 'hard' statements (from 'easy' to 'hard': high-validity robot: 83% → 91%, low-validity robot: 83% → 91%, neutral robot: 83% → 92%). This finding suggests that participants allocated more cognitive effort and processing time to read and understand more difficult statements. Apart from that, the most significant transitional pattern was a loop between the robot and classification categories. In contrast to the first pattern, transitions from the robot to the classification AOI diminished during 'hard' statements (from 'easy' to 'hard': high-validity robot: 21% → 16%, low-validity robot: 22% → 17%, neutral robot: 17% → 13%). Such a decrease could imply a strategic trade-off, where participants made more use of social or referential cues from the robot in 'easier' trials, while they relied more heavily on prolonged fixations on the statement assumptions during 'hard' tasks. In addition, this demonstrates a strategic shift towards more sustained fixations during more complex statements, rather than assessing the robotic cues.

Figure 14.

AOI Transition Probabilities for the Neutral Robot (Carl) in Easy and Hard Scenarios, highlighting the Gaze Transition Dynamics for Participants in the Experimental Game.



Note. Matrices show the likelihood of gaze moving from a starting AOI (y-axis) to an AOI destination (x-axis).

Besides the visualized differences, a Chi-Squared test was applied to formally assess whether participants' gaze strategies varied across the experimental conditions. The analysis revealed a significant effect of both difficulty level ($\chi(10)^2 = 38782.38$, $p < .001$) and robot identity ($\chi(16)^2 = 38721.37$, $p < .001$), confirming that participants altered their visual gaze strategy in response to task complexity and robot.

A post-hoc analysis using standardized residuals found that participants adapted a more focused strategy during 'hard' trials as the 'Cards → Cards' self-transition occurred significantly more frequently than expected (Residual = +53.61). In contrast, 'easy' statements were characterized by more social monitoring as the 'Classification → Robot' transition occurred significantly more often (Residual = +18.28). Examining the robot identity, our analysis showed a "stickier" gaze behavior towards the neutral robot as the 'Robot → Robot' self-transition showed significantly more trials (Residual = +63.29). Conversely, participants displayed significantly more interaction between the 'Robot → Classification' loop when interacting with the high-validity (Residual = **+10.09**) and low-validity robots (Residual = **+10.83**). Hence, participants engaged significantly more in the 'Robot ↔ classification' loop and displayed a more integrative back-and-forth strategy when the robot displayed a kind of referential gaze, no matter if reliable or not.

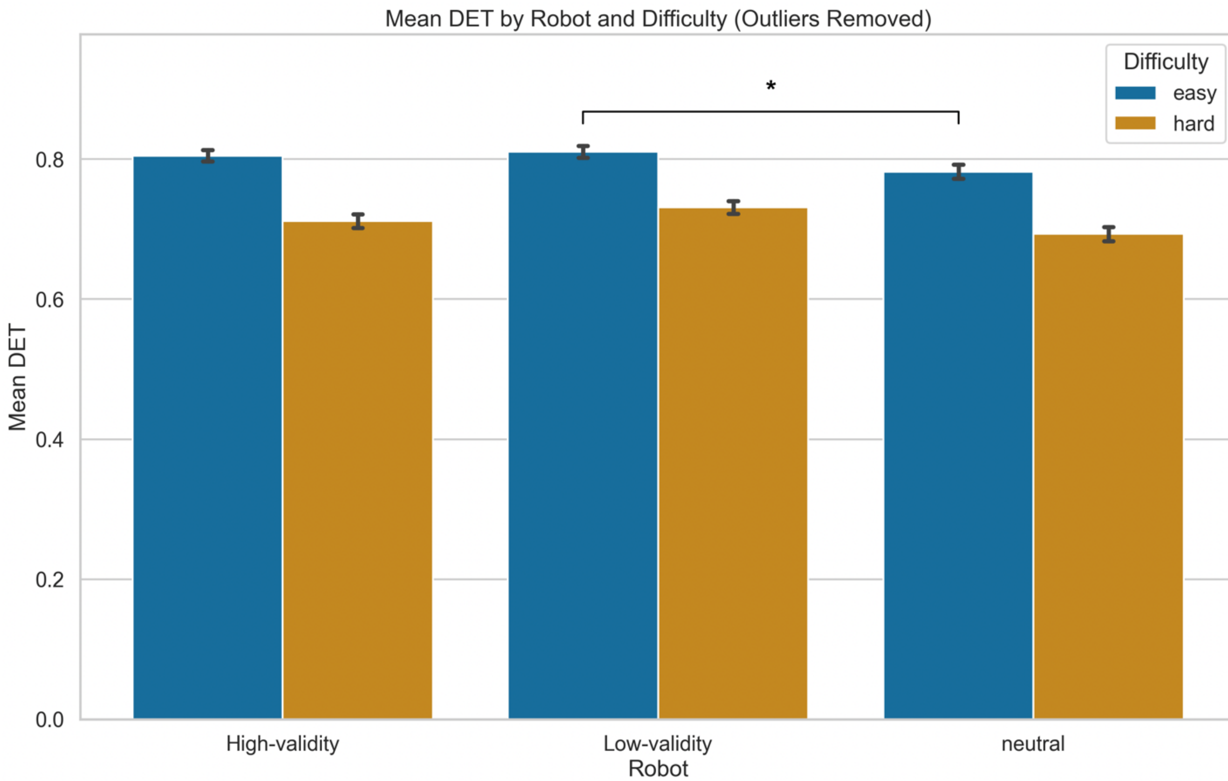
Recurrence Quantification Analysis

The previous transition analysis identified specific strategic moves and detailed gaze patterns applied by the participants. Upon this, the following analysis shifted to a macro-level understanding of the general gaze strategy. Particularly, the Recurrence Quantification Analysis (RQA) assessed the broader structure and the predictability of the entire strategic sequence for each trial. The RQA primarily focused on the metric of Determinism, which states how predictable and structured the participants' gaze strategy is (Anderson et al., 2013). A high Determinism score indicates a structured and more similar gaze strategy, while a low Determinism score would suggest a more chaotic and exploratory gaze path. Determinism rates were analyzed using a 3 (Robot: High-validity, low-validity, neutral) x 2 (Difficulty: Easy, hard) repeated-measures ANOVA. Before analysis, 1.76% (n = 50) of the data was removed based on the previously defined 2.5SD rule.

The ANOVA revealed a significant main effect of robot identity ($F(2, 62) = 5.79$, $p = .005$, $\eta g^2 = .009$) and difficulty ($F(1, 31) = 72.20$, $p < .001$, $\eta g^2 = .091$). No significant interaction effect was observed ($p = .909$). Post-hoc comparisons (with Bonferroni correction) revealed that gaze patterns in the presence of the low-validity robot ($M = 0.77$, $SD = 0.19$) were significantly more deterministic compared to the neutral robot ($M = 0.74$, $SD = 0.22$; Mean Diff = 0.03, 95% CI [0.01, 0.05], $p = .012$). No significant differences emerged between the high-validity and low-validity robots ($p = .065$), nor between the high-validity and neutral robots ($p = .360$). As visualized in Figure 15, the results demonstrate apparent differences in determinism according to the complexity level, but also represent the neutral robot with the lowest mean results, suggesting that participants adopted a marginally more exploratory approach for these and more difficult trials.

Figure 15.

Bar Chart of the Mean Determinism (DET) Scores for the Three Robots and Difficulty Levels.



Note. A high DET indicates higher predictability of gaze patterns. Data points identified as outliers were removed prior to analysis. Error bars represent ±1 standard error of the mean (SEM). *$p < .05$.

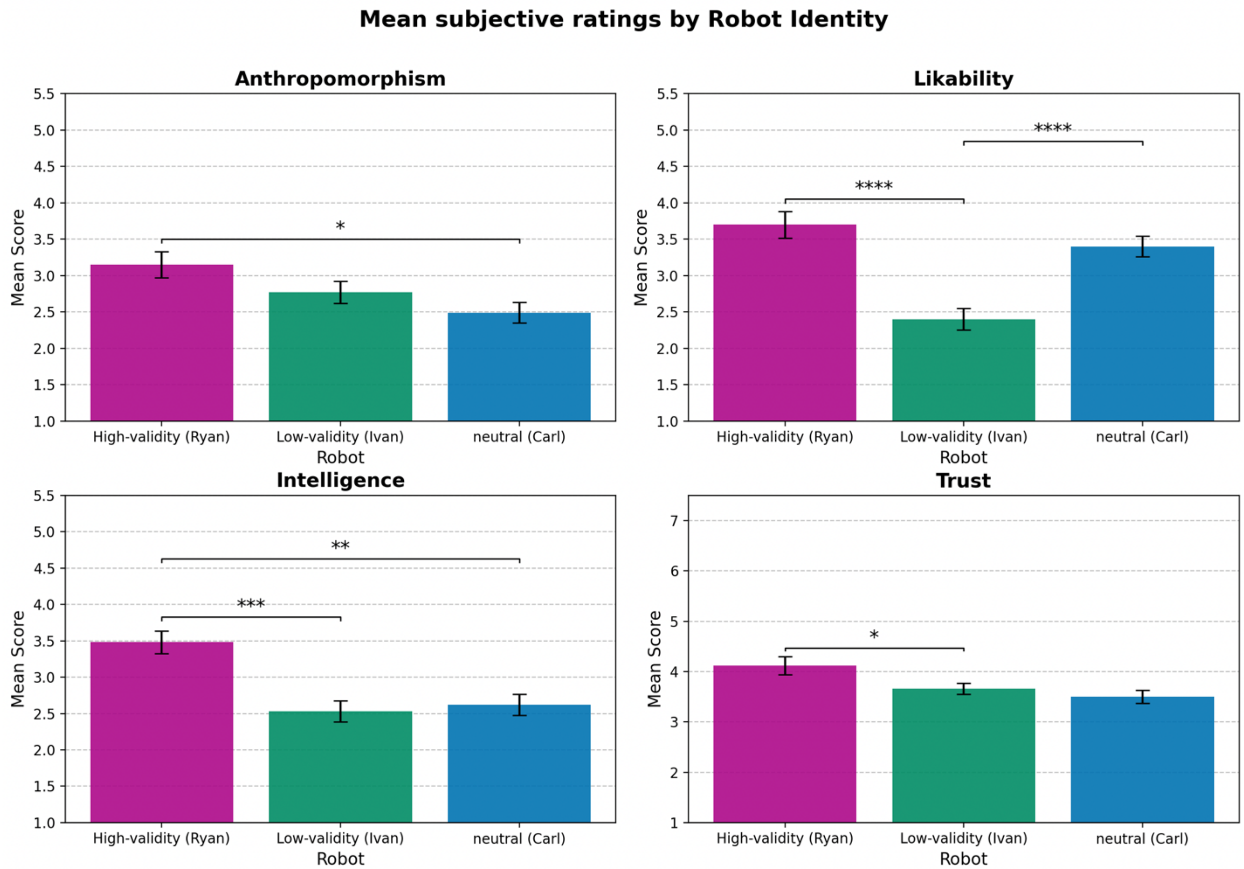
Analysis of Subjective Ratings

Across all subjective assessments from the Qualtrics Questionnaire, the high-validity robots' performance consistently surpassed that of the other two. As visualized in Figure 16, the high-validity robot (Ryan) obtained the highest ratings in anthropomorphism (M = 3.15, SD = 1.02), likability (M = 3.70, SD = 1.05), intelligence (M = 3.48, SD = 0.91), and trust (M = 4.12, SD = 1.03). The neutral robot (Carl) displayed the lowest average ratings for anthropomorphism

($M = 2.49$, $SD = 0.83$) and trust ($M = 3.50$, $SD = 0.74$), while the low-validity robot (Ivan) was the weakest for likability ($M = 2.40$, $SD = 0.84$) and intelligence ($M = 2.53$, $SD = 0.84$).

Figure 16.

Bar charts of the Average Ratings for Anthropomorphism, Likability, Intelligence, and Trust for each of the three Robots.



Note. Error bars represent the 95% confidence interval of the mean. $p < .05$, ** $p < .01$, *** $p < .001$, **** $p < .0001$.

In terms of reliability, the internal consistency of the assessment scales for each robot was evaluated using Cronbach's Alpha. Overall, all scales demonstrated acceptable to excellent internal consistency. All scales across the high-validity robot demonstrated excellent reliability, ranging from $\alpha = .895$ for trust to $\alpha = .951$. The intelligence ratings for the low-validity robot showed the lowest value of $\alpha = .787$, which can still be considered between acceptable and good.

To compare participants' subjective ratings of the robots, a series of one-way repeated measures ANOVAs was conducted across the four scales. In addition to the previously tested normality, Mauchly's test to check equality of the variances of the differences between all pairs of conditions was examined.

For the first scale, Anthropomorphism, Mauchly's test showed no violations of sphericity (W=.962, p=.545). A repeated-measures ANOVA revealed significant differences between the robots ($F(2,64) = 4.98$, $p = .010$, $\eta g^2 = .083$). Post-hoc comparisons using Bonferroni correction showed the high-validity robot ($M = 3.15$, $SD = 1.02$) rated significantly higher on anthropomorphism compared to the neutral robot ($M = 2.49$, $SD = 0.83$; Mean Diff = 0.66, 95% CI [0.19, 1.13], $p = .021$). No significant differences were found between low-validity ($M = 2.77$) and the neutral robot ($p = .531$), or between the high-validity and low-validity robots ($p = .184$).

For likability, a significant main effect across the robots was found ($F(2,64) = 17.07$, $p < .001$, $\eta g^2 = .281$). While Mauchly's test indicated a violation of sphericity ($W = .810$, $p = .038$), a Greenhouse-Geisser estimate of sphericity was used to correct the degrees of freedom ($\epsilon = .84$). The corrected ANOVA result also showed a main effect ($F(1.68, 53.79) = 17.07$, $p < .001$). Consequently, post-hoc comparisons showed the high-validity robot ($M = 3.70$, $SD = 1.05$) being significantly more likable than the low-validity one ($M = 2.40$, $SD = 0.84$; Mean Diff = 1.30, 95% CI [0.78, 1.82], $p < .001$). Similarly, the neutral robot (M = 3.40, $SD = 0.81$) showed significantly more likability compared to the low-validity robot (Mean Diff = 1.00, 95% CI

[0.64, 1.36], $p < .001$). No significant differences emerged between the high-validity and neutral robot ($p = .750$).

When it comes to perceived intelligence, a significant main effect of robot was observed during ANOVA testing ($F(2,64) = 11.49$, $p < .001$, $\eta g^2 = .202$). However, Mauchly's test indicated a violation of sphericity ($W = 0.794$, $p = .028$), leading to a Greenhouse-Geisser estimate of sphericity to correct the degrees of freedom ($\epsilon = 0.83$). The corrected ANOVA showed a significant main effect as well ($F(1.66, 53.08) = 11.49$, $p < .001$). Post-hoc comparisons showed that the high-validity robot ($M = 3.48$, $SD = 0.91$) was rated significantly higher on intelligence than both the neutral robot ($M = 2.62$, $SD = 0.84$; Mean Diff = 0.86, 95% CI [0.33, 1.39], $p = .007$), and the low-validity robot ($M = 2.53$, $SD = 0.84$; Mean Diff = 0.95, 95% CI [0.52, 1.38], $p < .001$). However, no significant difference emerged between neutral and low-validity robots ($p = 1.00$).

A similar pattern emerged for trust scores as significant differences emerged during ANOVA testing ($F(2, 64) = 5.59$, $p = .006$, $\eta g^2 = .096$). Next, Mauchly's test indicated a violation of sphericity ($W = 0.326$, $p < .001$), leading to the Greenhouse Geisser correction ($\epsilon = .60$). The ANOVA revealed a significant main effect of robot identity ($F(1.19,38.23) = 5.59$, $p = .018$). Consequently, post-hoc analysis showed that participants trusted the high-validity robot ($M = 4.12$) significantly more than the low-validity robot ($M = 3.66$, $SD = 0.63$; Mean Diff = 0.46, 95% CI [0.14, 0.78], $p = .019$). No significant differences were found in trust ratings between the neutral robot ($M = 3.50$) and the high-validity robot ($p = .069$), nor between the neutral and low-validity robot ($p = .780$).

Discussion

The current study aimed to investigate the influence of joint attention in Human-Robot Interaction (HRI), particularly the social mechanism of initiating joint attention from a screen-based robot towards a human. The conducted experiment accounted for the unique consideration of various independent factors such as complexity variation and gaze reliability, allowing a more dynamic and complex design. Our findings indicate that participants strategically adapted to the robot's reliability, developing a clear bias to trust and follow the gaze of the high-validity robot, while ignoring the low-validity one.

Performance Impact

The first hypothesis of this work examined the gaze impact on task performance, claiming that "Participants will perform significantly better in interaction with the high-validity robot". The current study findings partly contradict this hypothesis, as participants did not display a significantly higher or different score across one of the other robots. However, significant time differences during the classification process revealed that it took participants longer to make a decision when gaze cues were absent. In contrast to other conditions, participants were unable to devise a strategy and had to decide for themselves which statement was correct or incorrect. They had to rely on their knowledge and intuition.

The increased decision time in interaction with the neutral robot, which lacked external gaze hints, aligns with foundational literature. Mehlmann et al. (2014) found that the execution of referential gaze made a collaborative task twice as fast and significantly reduced errors. Similarly, Staudte & Crocker (2011) demonstrated that congruent gaze cues provide a clear performance benefit by speeding up comprehension. However, the same study introduced a 'benefit-disruption spectrum', stating that the robot's incongruent gaze cue demonstrated the slowest understanding times. Our results showed no significant time differences between the congruent (high-validity) and incongruent (low-validity) robots, and generally no differences in accuracy scores, which highlights a clear difference from the paper by Staudte and Crocker (2011). This difference can likely be attributed to the experimental design, as they used a single robot that applied congruent and incongruent gaze executions, while the current research used multiple robots. Consequently, it became easier for participants in our experiment to strategically trust or distrust the differing robots, as discussed in the following paragraphs.

Impact and Persistence of Gaze Following

The second hypothesis formulated was that "Participants' strategic bias to follow the high-validity robot leads to a kind of 'automation bias', causing users to follow its suggestion even if they are incorrect". This assumption emphasized gaze-following behavior, referring to participants' alignment of side choice with the robotic hints from the high-validity and low-validity robots. Our analysis underscored a strong support for that hypothesis. Fundamentally, participants followed the high-validity robot for incorrect gaze cues in 34.2% of trials. As expected, this trend increased for more complex tasks as participants followed incorrect gaze hints from the high-validity robot in 57.1% of the hard statements. Our findings indicate a substantial level of trust in the reliable robot, enough to override a participant's judgment when the task becomes more complex. This behavior can be characterized as an example of automation bias, defined as the human tendency to over-rely on suggestions from automated systems (Skitka et al., 1999).

The finding that participants followed the misleading gaze cue from a generally reliable robot is supported by previously conducted research. A similar study design from Staudte and Crocker (2011) found that participants would 'correct' a factually true statement when the referential gaze of a reliable robot pointed to a conflicting direction. The findings of this study confirm this tendency to trust robots' nonverbal cues, even to the point of questioning objective facts. Further, this finding is supported by Admoni and Scassellati (2017), who highlighted the role of humans in interpreting the robot's gaze as a direct signal of intention and focus of attention. In addition, our study revealed that this bias increased by task complexity, with participants' reliance on the robot enhancing from 34.2% to 57.1% of gaze following. In line with previous research, this finding underscores that humans are more likely to offload cognitive effort to an automated partner when this partner is perceived as competent (Lee & See, 2004; Risko & Gilbert, 2016).

These findings can also be interpreted within the framework of top-down and bottom-up processing (Katsuki & Constantinidis, 2014). Hence, the increased gaze following during more complex tasks can be interpreted by considering the interplay between reflexive and strategic attention. The robot's referential gaze hints could act as a salient, bottom-up cue, which naturally triggers a reflexive tendency to follow. Conversely, participants may have used their knowledge of the robot's identity as a strategic top-down process to either inhibit or trust this reflex. During easier trials, participants arguably displayed lower cognitive effort for the primary task – classifying the statement – and could use remaining resources to suppress a bottom-up reflex to follow the robotic gaze when it was incorrect. In contrast, for more complex statements, participants required increasing levels of mental effort in the primary task, which may have left fewer mental resources to suppress the bottom-up reflex to ignore the robotic gaze cue when it was misleading. This integrative approach, which allows for consideration of bottom-up and top-down processing, displays parallels with research conducted by Kompatsiari et al. (2018). In their study, mutual gaze was used to activate participants' engagement, which arguably increased the bottom-up urge of gaze following. Pursuing their argumentation, gaze-following behavior increased as the robots' eye contact strengthened participants' engagement and focus on the experimental game. Our findings extend this framework by suggesting that task complexity could be a key factor that is able to temporarily shift the balance from top-down towards more reflexive, bottom-up processing in human-robot interaction. Both studies highlight the dynamic interplay between bottom-up orienting and top-down control in shaping social attention.

In addition, our analysis considered gaze following behavior through the lens of signal detection theory (SDT). We employed the SDT framework to measure decision-making under uncertainty, particularly to measure an individual's ability to distinguish between signal and noise (Green & Swets, 1966). In the current context, SDT was applied to separate perceptual sensitivity from strategic bias in participants' decision-making process regarding the referential gaze applied. Our findings indicate a lower sensitivity (d') in the hard difficulty relative to the easy statements, suggesting that participants increasingly struggled with more complex tasks to judge if the referential gaze was correct. In other words, when the task became difficult, participants struggled to tell whether the robot was helping them or tricking them. Further, this score showed no differences between the high-validity and low-validity robots, indicating that participants were equally good at discriminating correct and incorrect gaze across the robots. More relevant in alignment with our assumption was an investigation of the response criterion (*c'*), which assesses participants' tendency and willingness to follow the robots. The results of this analysis represent one of the most critical findings of our research. As expected, participants tended to follow the high-validity robot and resist following the low-validity robot. However, this tendency also remained when the robots' gaze cues were misleading and incorrect. The participants simply 'stuck' with their previous strategy, no matter how difficult the statements got. Instead of trial-by-trial calculation, participants operated on a pre-established cognitive heuristic that was shaped by the robot's identity. Participants did not abandon their previously developed strategy in times of uncertainty. They strongly relied on it.

Participants' strong persistence in strategy deserves deeper reflection. Our findings provide an example of how humans interact with robots under uncertain conditions. The development of strong heuristics toward the high-validity and low-validity robot (e.g., "Ryan is helpful," "Ivan is not") became more relevant in complex and demanding tasks. As described by researchers like Kahneman (2011) and Gigerenzer and Gaissmaier (2011), this provides an example of how humans shift from analytical processing to more efficient heuristic-based strategies. In addition, humans are susceptible to the predictive validity of gaze cues. They are efficiently able to learn to inhibit reflexive orienting towards unreliable sources such as the low-validity robot (Friesen & Kingstone, 1998). As revealed in the Signal Detection Theory analysis, the observed automation bias was not a passive choice, but an active cognitive strategy powerful enough to override conflicting evidence.

Impact of Initiating Joint Attention on Gaze Predictability

The third hypothesis aimed to investigate gaze patterns and gaze strategies in depth. The hypothesis stated that "Participants will display more exploratory, unpredictable gaze behavior when interacting with the neutral robot, while the existence of referential gaze cues, albeit potentially incorrect, will lead to more predictable gaze patterns". The use of eye tracking analysis built a strong foundation to analyze not only basic features like dwell time but also to gain a deeper understanding of gaze strategies through advanced transition and recurrence analysis. Our analyses indicate a strong support for this hypothesis, showing that the determinism of gaze patterns was significantly lower when interacting with a robot that did not apply referential gaze. The absence of referential gaze and attentional guiding forced participants to adopt a more variable and exploratory search strategy. Further, this is supported by the AOI analysis, which reveals longer and more frequent visual attention towards the neutral robot. Taken together, the eye-tracking data represent a distinct trend: The presence of referential gaze, reliable or not, encourages participants to adopt a strategy during human-robot interaction, while its non-existence forces participants to adopt a more exploratory approach accompanied by a higher cognitive demand.

Previous work that included eye-tracking data provided evidence that the robot's referential gaze acts as a compelling guide for human attention. In other words, people automatically look where the robots look, even if the robotic cue is incorrect (Staudte & Crocker, 2011). Our recurrence analysis assigns a number to this effect as a significantly higher "determinism" score for gaze-referring robots proved a more predictable, structured pattern. These findings further align with established theories of visual attention, such as the "Guided Search" paradigm (Wolfe, 1994). From that perspective, the high-validity and low-validity robots provided a salient cue to 'guide' participants' search, which further simplifies the task and cognitive workload (Wolfe & Horowitz, 2017). Moreover, such a search strategy would be more structured, which explains its higher determinism scores. Conversely, the neutral robot represents an "unguided search", shaped by participants' cognitive load and less predictable gaze patterns (Liversedge & Findlay, 2000; Wolfe & Horowitz, 2017).

Our eye-tracking analysis also revealed an effect of participants displaying a "stickier" gaze towards the neutral robot. This was evidenced not only by more prolonged and more frequent dwell times, but also by a significantly higher probability of a 'Robot to Robot' self-transition. Given the robot's role as an active driver in conditions of referential gaze, we interpret the lingering human eye movement on the neutral robot as a behavioral marker for participants' uncertainty. Participants were naturally oriented towards the robot, expecting guidance from it. However, upon receiving no referential gaze, participants' gaze remained on the robot as they were forced to disengage from a simply reactive strategy and instead engage in a more cognitively demanding process. Eye-tracking literature indicates longer fixation durations as a primary indication of cognitive load or more difficult mental processing (Rayner, 2009).

Impact on perceived social attributes

Finally, the fourth hypothesis investigated the subjective ratings of interaction, stating that "The reliability of a robot's gaze will positively influence the self-reporting social attributes of likability, intelligence, anthropomorphism and trust". Analysis of the self-reporting questionnaires supported this assumption as the high-validity robot, Ryan, consistently received the most favorable ratings across all four measured attributes: anthropomorphism, likability, intelligence, and trust. This outcome supports the overarching effect of reliable referential gaze on social perception in human-robot interaction. It aligns with previous scientific research, linking context-aware gaze to robots being perceived as more natural, likable, and intelligent (Admoni & Scassellati, 2017).

However, post-hoc analyses revealed a more nuanced picture, adding value to the understanding of the importance of reliability in joint attention as well as the consequences of its absence. Pairwise comparisons displayed an unlikability or off-putting nature to the low-validity robot, whereas the high-validity and neutral robots did not show major preferences in terms of likability. While a trend towards the high-validity robot was identified, the non-significance compared to the neutral robot can be attributed to the robots' general limitations, as participants saw only a static picture with interactive eyes, which were also limited in their gaze application.

However, the significant differences remained stable across scores of anthropomorphism and intelligence, with the reliable robot showing higher ratings. Moreover, perceived intelligence was rated significantly higher for the high-validity robot compared to the other robots, which further strengthens literature insights that reliable referential gaze leads robots to appear more competent (Admoni & Scassellati, 2017).

Interestingly, this study found no significant differences across self-perceived trust scores. This finding contradicts previous scientific work, which frequently reported higher scores of robots that applied referential gaze (Mutlu et al., 2009). Counterintuitively, the non-significances in the self-reporting data also contrast with our behavioral findings, as the gaze-following and eye-tracking analysis revealed significant preferences for following the high-validity robot, shaped by more predictable strategies of trustworthiness. In consistency with our behavioral data and literature context, it’s plausible that the Trust-Perception Scale for HRI (TPS-HRI) from Schaefer (2016) could not adequately reflect participants’ trust level. Following the arguments for a participatory and context-aware approach (Korpan, 2024), a universal or generic trust scale might oversimplify nuanced ways in which trust is formed. Moreover, a scale that was validated primarily for a military simulation context may fail to capture the dynamic and social dimensions of joint attention (Korpan, 2024).

Strengths of the Study

Unlike previous studies, this research examines the mechanism of initiating joint attention in a more complex consideration, accounting for its reliability and its influence on task complexity. While using a 3 x 2 study design to examine joint attention with three robots in complexity-varying tasks, the robots implicitly varied in two key aspects: the presence of Initiating Joint Attention and the reliability of the gaze cues. This variation allowed us to investigate what is rarely reported in HRI literature: the consequences of incorrect application of referential gaze (Admoni & Scassellati, 2017). If humans increasingly interact with intelligent machines and robots, blind trust can lead to high costs or dangerous accidents, due to an over-reliance on the robot's indication (Parasuraman & Manzey, 2010). Our study addresses this gap, acknowledging the varying reliability of attentional cues.

In addition, a substantial strength of this study is the use of a within-subjects repeated-measures design. This experimental design allowed each participant to act as their own baseline, thereby comparing the three robots directly, which strengthens the internal validity of our conclusions. Further, the procedure for presenting the classification statements and robots was randomized. While still algorithmically accounting for the same amount of "easy" and "hard" statements for each of the three robots, the randomization procedure prevented confounding variables related to specific order or content effects (Shadish et al., 2002). Similarly, it ensured a fair and balanced comparison of the robots. Upon this, the construction of our experiment connected the experimental game via a local network to the robotic gaze system, which enabled robotic gaze cues to be linked directly and intentionally to the task. Even when the robots' gaze was unreliable, it was not randomly looking at one target, which makes an incorrect gaze cue also task relevant. In addition, the task was arguably more naturalistic compared to other research (Huang & Thomaz, 2011), while we still maintained experimental control.

Finally, our experiment provided a range of measurement metrics, allowing for increased validity. Using the self-constructed classification game and gaze control system, this research accounted for participants' performance and gaze-following data but additionally incorporated their gaze behavior through eye-tracking metrics. In addition, a post-questionnaire measured participants' self-reporting tendencies regarding the social attributes of the robots. Such methodological triangulation revealed more nuanced insights and provided a richer, more comprehensive understanding of human-robot interaction.

Future Directions

The findings of this study are considered in light of some methodological limitations, which in turn suggest valuable directions for future research. First, the experimental setup consisted of a static screen-based robot face with an interactive gaze system on a monitor. The robot was therefore neither an embodied agent nor was it particularly flexible in its facial movements. The lack of physical embodiment can be a significant consideration, as scientific literature suggests that people perceive and behave differently towards physically embodied robots compared to virtual agents, primarily with increased attention and stronger social engagement (Li, 2015). However, the use of screen-based robots in human-robot interaction literature is a standard and commonly used methodology, often to achieve high experimental control over variables such as gaze cues (Admoni & Scassellati, 2017). Despite that, future research could examine and validate our results using a physically embodied robot. Socially embodied robots such as “Furhat” (Furhat Robotics, n.d.) provide flexible and straightforward API connections, enabling the use of a similar gaze control system in humanoid robots. In addition, such studies could investigate the effects of embodiment, potentially strengthening or weakening the effects of joint attention in human-robot interaction.

A second limitation refers to the self-constructed gaze control system. Building a complete human-like gaze system, which perceives its environment and reacts accordingly, is still very limited (Admoni & Scassellati, 2017; Mishra & Skantze, 2022). Our system specifically focused on joint attention and perceived its environment only in a very limited way: by interpreting game events and the presence of human faces. Further, the execution of referential gaze was triggered to appear in a static timeframe after a card reveal automatically. While this is beneficial in terms of overview and controlled manipulation, it also limits the interactivity and reactivity of the robot in communication with humans, which is often a significant factor in engagement, natural movements, and social feelings towards the robot (Fong et al., 2003). Future studies might extend the gaze control system to become more flexible. One example of such direction is highlighted in the paper by Mishra and Skantze (2022), who developed a planned gaze control system, which plans the robot's gaze for a future, rolling time window instead of being purely reactive. Like Pereira et al (2019), their gaze system not only used a proactive layer for referential gaze, but also integrated a responsive layer to display responsive gaze. In alignment with the recent breakthroughs in areas of deep and reinforcement learning (LeCun et al., 2015), an additional exemplary approach might use not only a heuristically driven system, but instead build a combination or even a fully data-driven system.

Furthermore, the sample size was modest (n = 33) and consisted only of university students, decreasing the generalizability to a broader population. In consideration of the context that humanoid robots will interact in various fields with various people, further investigation could account for a larger sample size with different demographic characteristics.

Finally, additional review and investigation are needed to assess the bidirectionality of combining the mechanisms of responding and initiating joint attention, but also to assess them separately as we did. These two mechanisms of joint attention can be very different and may prove to be useful or less useful in different contexts. Thus, the last recommendation of this paper is to test the social mechanisms of joint attention in varying contexts and objectives. For example, on a production line, the versatile use of referential gaze is likely to be advantageous due to increased speed and fewer errors, whereas in school, reciting instructions could hinder independent learning.

Practical Implications

Beyond theoretical relevance, the results also offer insights for practical applications in the field of research and development of humanoid robots. One essential finding of this overall research for designers and engineers is the prioritizing of reliability when it comes to the implementation of referential gaze or initiating joint attention. Our results clearly indicate that a low-validity robot was not only seen as less capable, but it was also actively disliked and distrusted. For several social attributes, such as likability, intelligence, anthropomorphism, or trust, participants did not show any significant preferences towards the low-validity robot compared to the neutral robot, which did not apply any kind of referential gaze. However, a high-validity gaze behavior evoked social preferences and performance improvements. In other words, the practical implication of this research could be formulated in the manner of "Do it right or don't do it at all".

More critically, our research outcomes highlight how humans develop strategies and automation bias to trust humanoid robots. In a societal context, this underscores the particular risk to over-rely on machines and robotics, even to override one's own judgment. This research can be used to raise awareness of this automation bias and overreliance, particularly to treat human-robot interaction with caution in certain fields such as healthcare and education (Breazeal, 2003).

Conclusion

This research emphasized the role of reliable referential gaze in Human-Robot Interaction. Our central finding lies in the gaze-following analysis, which indicates humans' development of powerful strategic bias, learning to consistently trust a reliable robotic gaze, and even overriding their own judgment. Trust towards a high-validity robot has led to a kind of automation bias, causing participants to follow the robot's suggestions. This was particularly the case for more complex tasks, as participants arguably experienced a higher cognitive workload, leaving them with less mental capacity to judge the correctness of the robotic hints. Upon that, a more detailed look through the eye-tracking data confirmed participants' strategic approach, as the existence of referential gaze, albeit of its validity, revealed more structured and organized gaze patterns. The absence of referential gaze significantly altered participants' gaze behavior, showing less structured and more explorative gaze strategies. Despite that, participants spent more time looking towards the robot without the initiation of joint attention, arguably in expectation of receiving a gaze cue or reaction. Further, the eye-tracking data provided powerful support for participants' strategic development towards the robots that applied initiating joint attention. Hence, participants developed a more predictable gaze strategy when interacting with robots that execute referential gaze. For example, we observed participants switching their gaze between the robot and classification theory more frequently when the robot provided initiated joint attention. Despite many similarities in the eye-tracking data, particularly from a strategic and predictable nature, the reliability of gaze cues showed not only significant differences in the gaze following, but also in the self-reporting tendencies. Our results show that participants substantially preferred a robot that applied a reliable gaze compared to one with frequently misleading gaze hints. Notably for likability, the robot with unreliable gaze cues showed significantly worse values compared to both the reliable robot and the robot without referential gaze. Participants' tendency to trust the robot with a reliable gaze, as opposed to one that displays an unreliable gaze, is also supported by our behavioral data, as the gaze-following analysis reveals a clear preference for following the reliable, high-validity robot.

References

Admoni, H., Dragan, A., Scassellati, B., & Srinivasa, S. (2014). Deliberate delays during robot-to-human handovers improve compliance with gaze communication. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-Robot Interaction* (HRI '14). Association for Computing Machinery. https://doi.org/10.1145/2559636.2559682

Admoni, H., & Scassellati, B. (2017). Social eye gaze in human-robot interaction: A review. *Annual Review of Control, Robotics, and Autonomous Systems, 4*(1), 6.1-6.23. https://doi.org/10.5898/JHRI.6.1.Admoni

Anderson, N. C., Bischof, W. F., Laidlaw, K. E. W., Risko, E. F., & Kingstone, A. (2013). Recurrence quantification analysis of eye movements. *Behavior Research Methods*, *45*(3), 842–856. https://doi.org/10.3758/s13428-012-0299-5

Andrist, S., Gleicher, M., & Mutlu, B. (2017). Looking coordinated: Bidirectional gaze mechanisms for collaborative interaction with virtual characters. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. Association for Computing Machinery. https://doi.org/10.1145/3025453.3026033

Bartneck, C., Kulić, D., Croft, E., & Zoghbi, S. (2009). Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots. *International Journal of Social Robotics*, *1*(1), 71–81. https://doi.org/10.1007/s12369-008-0001-3

Bayliss, A. P., Murphy, E., Naughtin, C. K., Kritikos, A., Schilbach, L., & Becker, S. I. (2013). Gaze leading: Initiating simulated joint attention influences eye movements and choice behaviour. *Journal of Experimental Psychology: General*, *142*(1), 76–92. https://doi.org/10.1037/A0029286

Boston Dynamics. (n.d.). *Atlas*. Retrieved from https://bostondynamics.com/atlas/

Breazeal, C. (2003). Toward sociable robots. *Robotics and Autonomous Systems, 42*(3-4), 167–175. https://doi.org/10.1016/S0921-8890(02)00373-1

Charman, T. (2000). Theory of mind and the early diagnosis of autism. In S. Baron-Cohen, H. Tager-Flusberg, & D. J. Cohen (Eds.), *Understanding other minds: Perspectives from developmental cognitive neuroscience* (2nd ed., pp. 422–441). Oxford University Press.

Chen, J. Y. C., & Barnes, M. J. (2014). Human–agent teaming for multirobot control: A review of human–agent teaming research. *Human Factors: The Journal of the Human Factors and Ergonomics Society, 56*(2), 293-315. https://doi.org/10.1109/THMS.2013.2293535

Emery, N. J. (2000). The eyes have it: The neuroethology, function, and evolution of social gaze. *Annual Review of Neuroscience, 23*(1), 527–563. https://doi.org/10.1016/S0149-7634(00)00025-7

Fong, T., Nourbakhsh, I., & Dautenhahn, K. (2003). A survey of socially interactive robots. *Robotics and Autonomous Systems, 42*(3-4), 143–166. https://doi.org/10.1016/S0921-8890(02)00372-X

Friesen, C. K., & Kingstone, A. (1998). The eyes have it! Reflexive orienting is triggered by nonpredictive gaze. *Psychonomic Bulletin & Review*, *5*(3), 490–495.

Frischen, A., Bayliss, A. P., & Tipper, S. P. (2007). *Gaze cueing of attention: Visual attention, social cognition, and individual differences.* Psychological Bulletin, 133(4), 694–724. https://doi.org/10.1037/0033-2909.133.4.694

Furhat Robotics. (n.d.). *Furhat Robotics*. Retrieved June 7, 2025, from https://furhatrobotics.com/

Gigerenzer, G., & Gaissmaier, W. (2011). Heuristic decision making. *Annual Review of Psychology*, *62*(1), 451–482. http://dx.doi.org/10.1146/annurev-psych-120709-145346

Green, D. M., & Swets, J. A. (1966). *Signal detection theory and psychophysics*. Wiley.

Goddard, K., Roudsari, A., & Wyatt, J. C. (2012). Automation bias: a systematic review of frequency, effect mediators, and mitigators. *Journal of the American Medical Informatics Association: JAMIA*, *19*(1), 121–127. https://doi.org/10.1136/amiajnl-2011-000089

Hoffmann, G., & Breazeal, C. (2004). Collaboration in human-robot teams. *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, 1167-1170. https://doi.org/10.2514/6.2004-6434

Huang, C.-M., & Thomaz, A. L. (2011). *Effects of responding to, initiating and ensuring joint attention in human-robot interaction*. In 2011 RO-MAN: The 20th IEEE International Symposium on Robot and Human Interactive Communication (pp. 65–70). IEEE. https://doi.org/10.1109/ROMAN.2011.6005230

Imai, M., Ono, T., & Ishiguro, H. (2003). Physical relation and expression: Joint attention for human-robot interaction. *IEEE Transactions on Industrial Electronics, 50*(4), 636–643. https://doi.org/10.1109/TIE.2003.814769

Kahneman, D. (2011). *Thinking, fast and slow*. Farrar, Straus and Giroux.

Kanda, T., Hirano, T., Eaton, D., & Ishiguro, H. (2004). Interactive robots as social partners and peer tutors for children: A field trial. *Human-Computer Interaction, 19*(1-2), 61–84. https://doi.org/10.1207/s15327051hci1901&2_4

Katsuki, F., & Constantinidis, C. (2014). Bottom-up and top-down attention: different processes and overlapping neural systems: Different processes and overlapping neural systems. *The Neuroscientist: A Review Journal Bringing Neurobiology, Neurology and Psychiatry*, *20*(5), 509–521. https://doi.org/10.1177/1073858413514136

Kleinke, C. L. (1986). Gaze and eye contact: A research review. *Psychological Bulletin, 100*(1), 78–100. https://doi.org/10.1037/0033-2909.100.1.78

Kompatsiari, K., Ciardo, F., Tikhanoff, V., Metta, G., & Wykowska, A. (2018). On the role of eye contact in gaze cueing. *Scientific Reports*, *8*(1), 17842. https://doi.org/10.1038/s41598-018-36136-2

Korpan, R. (2024). Towards a participatory and social justice-oriented measure of human-robot trust. In *arXiv [cs.RO]*. http://arxiv.org/abs/2402.15671

Land, M. F., & Hayhoe, M. (2001). In what ways do eye movements contribute to everyday activities? *Vision Research*, *41*(25–26), 3559–3565. https://doi.org/10.1016/s0042-6989(01)00102-x

Lavie, N. (2005). Distraction, task difficulty, and focused attention: A perceptual load perspective. *Trends in Cognitive Sciences, 9*(2), 75–82. https://doi.org/10.1016/j.tics.2004.12.008

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lee, J. D., & See, K. A. (2004). Trust in automation: Designing for appropriate reliance. *Human Factors*, *46*(1), 50–80. https://doi.org/10.1518/hfes.46.1.50_30392

Lemaignan, S., Warnier, M., Sisbot, E. A., Clodic, A., & Alami, R. (2017). Artificial cognition for social human–robot interaction: An implementation. *Frontiers in Robotics and AI, 4*. https://doi.org/10.1016/j.tics.2004.12.004

Li, J. (2015). The benefit of being physically present: A meta-analysis of the physical embodiment effect in human-robot interaction. In *Proceedings of the 24th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)* (pp. 124-131). IEEE. http://dx.doi.org/10.1016/j.ijhcs.2015.01.001

Liversedge, S. P., & Findlay, J. M. (2000). Saccadic eye movements and cognition. *Trends in Cognitive Sciences*, *4*(1), 6–14. https://doi.org/10.1016/S1364-6613(99)01428-7

L1Fthrasir. (2023). Facts-true-falsts- [Data set]. Hugging Face. https://huggingface.co/datasets/L1Fthrasir/Facts-true-false

Mehlmann, G., Häring, M., Janowski, K., Baur, T., Gebhard, P., & André, E. (2014). Exploring a model of gaze for grounding in multimodal hri. In Proceedings of the 16th international conference on multimodal interaction (pp. 247–254). https://doi.org/10.1145/2663204.2663275

Metta, G., Natale, L., Nori, F., Sandini, G., Vernon, D., Fadiga, L., et al. (2010). The iCub humanoid robot: an open-systems platform for research in cognitive development. *Neural Netw.* 23, 1125–1134. https://doi.org/10.1016/j.neunet.2010.08.010

Mishra, C., & Skantze, G. (2022). Knowing where to look: A planning-based architecture to automate the gaze behavior of social robots. In *Proceedings of the 31st IEEE International Conference on Robot & Human Interactive Communication (RO-MAN 2022)* (pp. 481-488). IEEE. https://doi.org/10.1109/RO-MAN53752.2022.9900740

Mundy, P. (2018). A review of joint attention and social-cognitive brain systems in typical development and autism spectrum disorder. *European Journal of Neuroscience, 47*(6), 497–514. https://doi.org/10.1111/ejn.13720

Mundy, P., & Newell, L. (2007). Attention, joint attention, and social cognition. *Current Directions in Psychological Science, 16*(5), 269–274. https://doi.org/10.1111/j.1467-8721.2007.00518.x

Mutlu, B., Shiwa, T., Kanda, T., Ishida, T., & Hagita, N. (2009). Footing in human-robot conversations: how robot gaze affects participant ratings and conversational floor. *Proceedings of the 4th ACM/IEEE international conference on Human-robot interaction*, 61-68. https://doi.org/10.1145/1514095.1514109

OpenAI. (2025). ChatGPT (DALL·E 3 version) [Large language model]. https://chatgpt.com

Pan, M. K. X. J., Choi, S., Kennedy, J., McIntosh, K., Zamora, D. C., Niemeyer, G., Kim, J., Wieland, A., & Christensen, D. (2020). *Realistic and interactive robot gaze*. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 11072–11078). IEEE. https://doi.org/10.1109/IROS45743.2020.9341297

, M. K. X. J., Choi, S., Kennedy, J., McIntosh, K., Zamora, D. C., Niemeyer, G., Kim, J., Wieland, A., & Christensen, D. (2020). *Realistic and interactive robot gaze*. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 11072–11078). IEEE. https://doi.org/10.1109/IROS45743.2020.9341297

Parasuraman, R., & Manzey, D. H. (2010). Complacency and bias in human use of automation: An attentional integration. *Human Factors*, *52*(3), 381–410. https://doi.org/10.1177/0018720810376055

Pereira, A., Oertel, C., Fermoselle, L., Mendelson, J., & Gustafson, J. (2019). Responsive joint attention in human-robot interaction. In Proceedings of the 2019 ieee/rsj international conference on intelligent robots and systems (iros) (pp. 1080–1087). IEEE.https://doi.org/10.1109/IROS40897.2019.8968130

Pfeiffer, U. J., Vogeley, K., & Schilbach, L. (2013). From gaze cueing to dual eye-tracking: Novel approaches to investigate the neural correlates of gaze in social interaction. *Neuroscience & Biobehavioral Reviews, 37*(10 Pt 2), 2516–2528. https://doi.org/10.1016/j.neubiorev.2013.07.017

Posner, M. I. (1980). Orienting of attention. *The Quarterly Journal of Experimental Psychology, 32*(1), 3–25. https://doi.org/10.1080/00335558008248231

Qualtrics. (2025). Qualtrics [Software]. https://www.qualtrics.com

Rayner, K. (2009). Eye movements and attention in reading, scene perception, and visual search. *Quarterly Journal of Experimental Psychology*, *62*(8), 1457–1506. https://doi.org/10.1080/17470210902816461

Risko, E. F., & Gilbert, S. J. (2016). Cognitive offloading. *Trends in Cognitive Sciences*, *20*(9), 676–688. https://doi.org/10.1016/j.tics.2016.07.002

Scassellati, B. (2002). *Theory of mind for a humanoid robot* [Doctoral dissertation, Massachusetts Institute of Technology]. MIT Libraries. https://dspace.mit.edu/handle/1721.1/8381

Schaefer, K. E. (2016). *Measuring trust in human-robot interactions: Development of the trust perception scale-HRI*. In *Proceedings of the 11th ACM/IEEE International Conference on Human-Robot Interaction – Trust Workshop*. http://dx.doi.org/10.1007/978-1-4899-7668-0_10

Shadish, W. R., Cook, T. D., & Campbell, D. T. (2002). *Experimental and quasi-experimental designs for generalized causal inference.* Houghton, Mifflin and Company

Skitka, L. J., Mosier, K. L., & Burdick, M. D. (1999). Does automation bias decision-making? *International Journal of Human-Computer Studies*, *51*(5), 991–1006. https://doi.org/10.1006/ijhc.1999.0252

Staudte, M., & Crocker, M. W. (2011). Investigating joint attention mechanisms through spoken human–robot interaction. *Cognition, 120*(2), 268–291. https://doi.org/10.1016/j.cognition.2011.05.005

Tesla. (n.d.). *Optimus*. Retrieved from https://www.tesla.com/en_eu/AI

Tomasello, M. (1999). *The cultural origins of human cognition*. Harvard University Press.

Willemse, C., Marchesi, S., & Wykowska, A. (2018). Robot faces that follow gaze facilitate attentional engagement and increase their likeability. *Frontiers in Psychology, 9*, Article 70. https://doi.org/10.3389/fpsyg.2018.00070

Wolfe, J. M. (1994). Guided Search 2.0: A revised model of visual search. *Psychonomic Bulletin & Review*, *1*(2), 202–238. https://doi.org/10.3758/BF03200774

Wolfe, J. M., & Horowitz, T. S. (2017). Five factors that guide attention in visual search. *Nature Human Behaviour*, *1*(3), 0058. https://doi.org/10.1038/s41562-017-0058

Zhou, Y., Wang, D., Yu, Y., & Zhang, Z. (2023). Vision-based static gesture recognition for human–computer interaction. *Electronics*, *12*(13), 2805. https://doi.org/10.3390/electronics12132805

Appendixes.

Appendix 1.

Prompting to generate the three static pictures for a robots face that account for each of the three different gaze conditions in the gaze control system in OpenAI's image-generation model DALL E 2 via the GPT-4 console.

<u>Primary prompt for the robot with high reliability gaze:</u>

- "Create a 3D picture of a realistic, friendly humanoid robot that looks directly at the camera with a straight face. The robot should display a gentle, approachable expression. Further, it has a smooth, rounded face with large, expressive, and realistic eyes that convey a sense of curiosity. The neck is exposed and contrasts with the smooth face, revealing intricate black and grey mechanical joints and wiring. The overall design should show strong similarities to the iCub robot. The overall picture should display the robots face, his neck and partly his shoulders"

<u>Further prompting for the low-reliability robot and control robot:</u>

- "Based on the previous picture, please generate additional 3D pictures of similar-looking robots, that display the same gentle, approachable expression with a smooth, rounded face with large, expressive and realistic eyes and similarities to the iCub robot. The robotic shape and its expression should be similar to the previous picture, but the robot should look differently. Imagine a scenario where this robot could be a cousin or another relative of the previous robot. "
- "Generate an alternative picture based on the previously used prompt"

Appendix 2.

Codebase for the Gaze Control System and Experimental Game.

Valid link: https://github.com/Devin037/Bachelor-Thesis/tree/main/experiment

Appendix 3.

Barcode for Data Analysis as well as Data Cleaning and Data Transformation in Python and R.

Barcode for Scripts of Data Preprocessing and Data Cleaning before actual data analysis:

Valid link: https://github.com/Devin037/Bachelor-Thesis/tree/main/data-transformation-and-cleaning

Barcodes for Data Analyses Scripts in R for the analyses of performance, gaze following, Qualtrics questionnaire and for the Python Scripts for the Eye-tracking Analyses:

Python Scripts (Eye-tracking)

https://github.com/Devin037/Bachelor-Thesis/tree/main/data-analysis/python-eye-tracking

R-Scripts:

https://github.com/Devin037/Bachelor-Thesis/tree/main/data-analysis/R

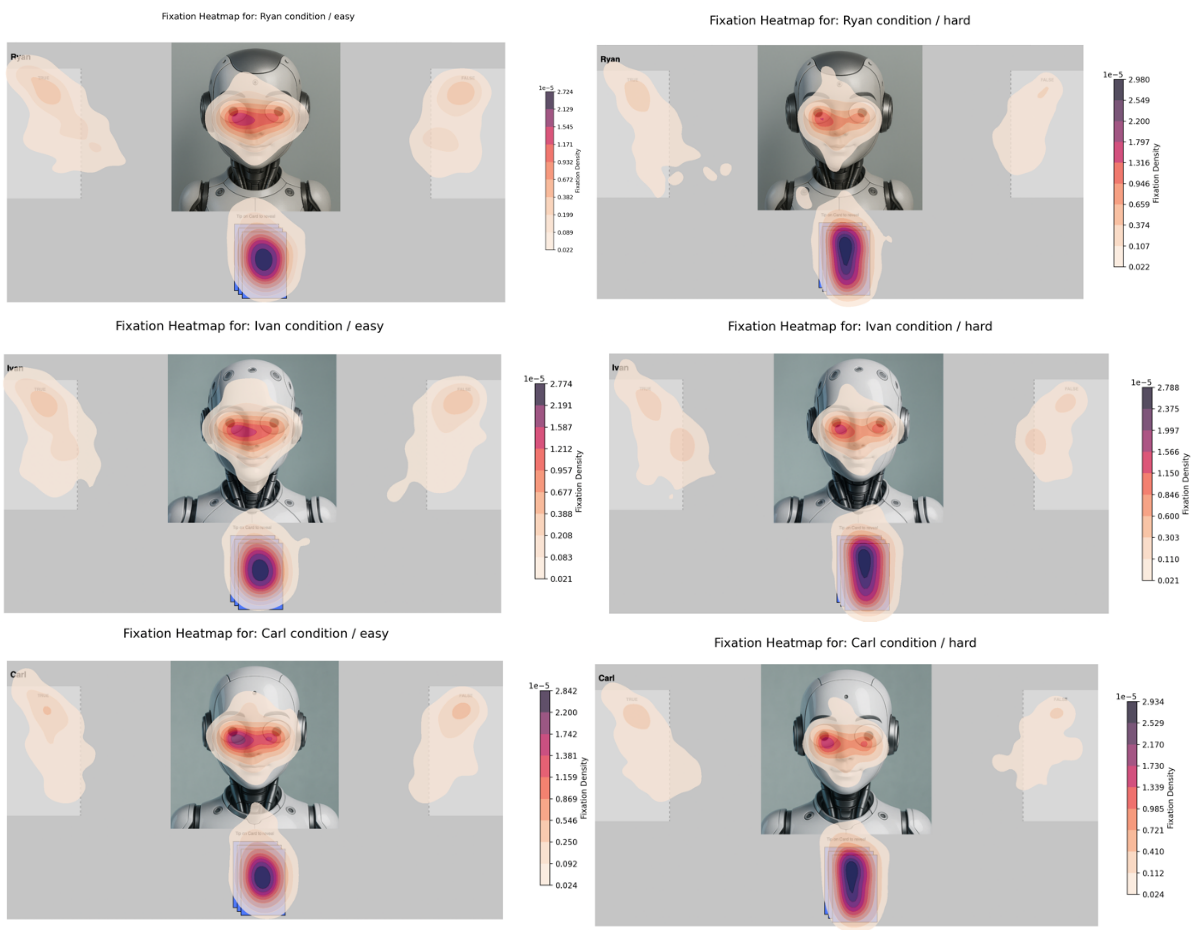
Note. The raw code of data cleaning, transformation and analysis can also be found in Appendix 7.

Appendix 4.

Heatmaps and advanced AOI transition maps for each robotic and difficulty condition as additional data of the Results Section next to the shown visualizations.

Figure 1.

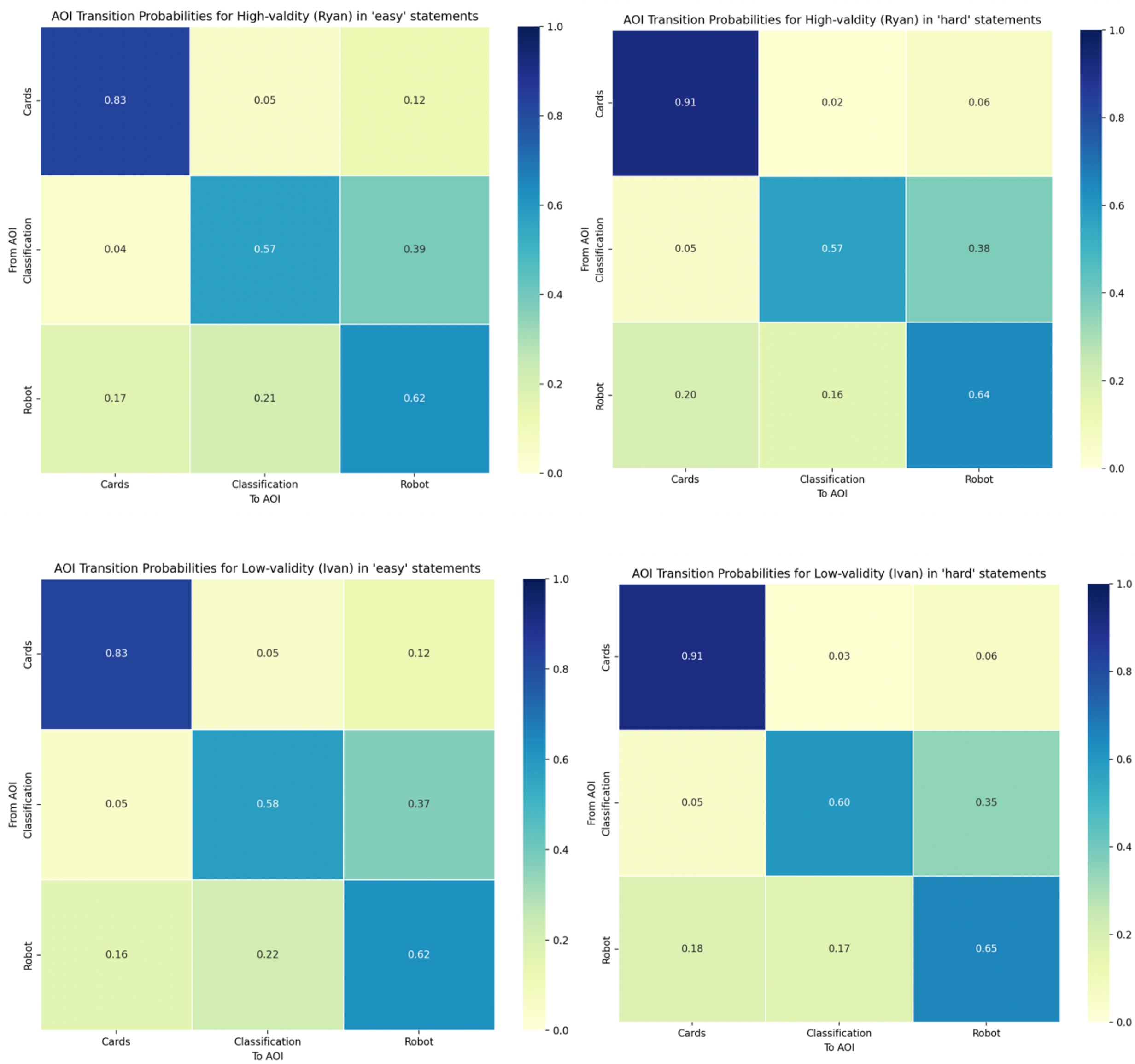
Heatmaps of each robotic condition for "easy" and "hard" difficulties

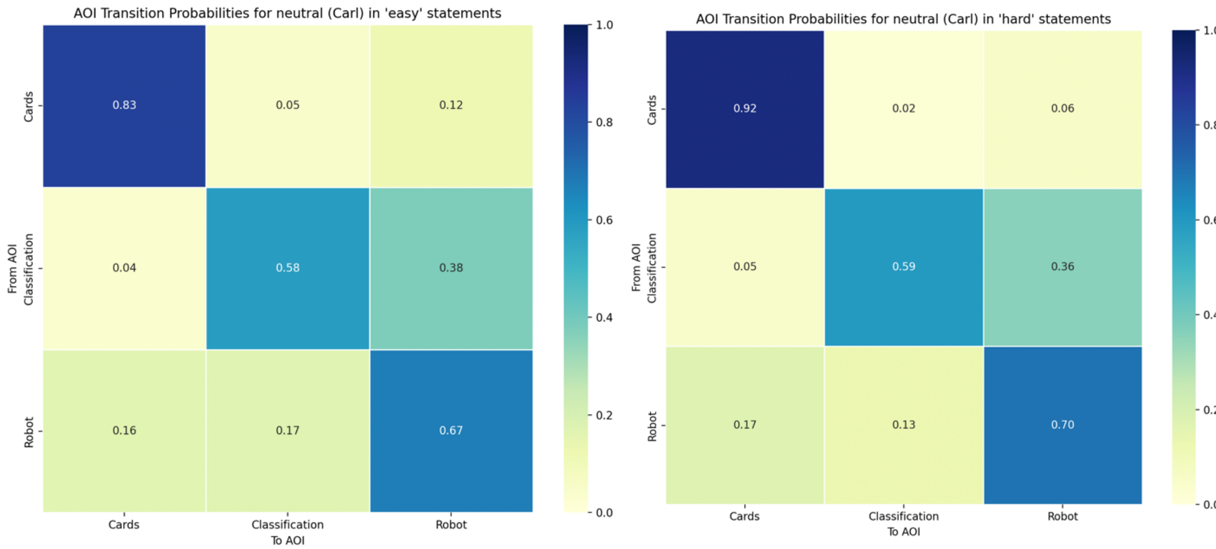


Note. The row with the first two pictures shows the high-validity robot (Ryan) for both categories ('easy' on the left side, 'hard' on the right side). The second row shows two pictures for the low-validity Robot (Ivan) with the easy category on the left and hard on the right side. The third row shows the heatmaps for the neutral robot (Carl), also with easy statements on the left and hard statements on the right side.

Figure 2.

Advanced AOI maps for each robot and difficulty level, starting with the high-validity Robot (Ryan) in the first row, followed by the low-validity Robot (Ivan) in the second row.





Note. Pictures on the left side refer to the "easy" category, while pictures on the right side can be marked as "hard".

Appendix 5.

Table of the questions used for the classification game.

Round	type	difficulty	category	question	correct_answer
default	boolean	easy	General Knowledge	Move statement to one of the sides to start the actual game!	TRUE
1	boolean	easy	General Knowledge	The Sun is more massive than earth	TRUE
1	boolean	easy	General Knowledge	The Eiffel Tower is located in Paris France.	TRUE
1	boolean	easy	General Knowledge	The fastest fish in the world is the goldfish	FALSE
1	boolean	easy	General Knowledge	French is an official language in Canada.	TRUE
1	boolean	easy	General Knowledge	Ananas is mostly used as the word for Pineapple in other languages.	TRUE
1	boolean	easy	General Knowledge	The color orange is named after the fruit.	TRUE
1	boolean	easy	General Knowledge	Mount Everest is the highest mountain in the world	TRUE
1	boolean	easy	General Knowledge	Earth has multiple moons	FALSE
1	boolean	easy	General Knowledge	The Sun rises from the North.	FALSE
1	boolean	easy	General Knowledge	Coral reefs are located underwater.	TRUE
1	boolean	hard	General Knowledge	The respiratory system prevents the exchange of gases between the body and the environment	TRUE
1	boolean	hard	General Knowledge	The smallest volcano in the world is located in Hawaii.	FALSE
1	boolean	hard	General Knowledge	Light can exhibit neither wave-like nor particle-like properties.	FALSE
1	boolean	hard	General Knowledge	The electron configuration of an atom determines its physical properties.	FALSE
1	boolean	hard	General Knowledge	The Doppler effect causes the change in frequency or wavelength of a wave in relation to an observer	TRUE
1	boolean	hard	General Knowledge	The first successful human heart transplant was performed in 1967	TRUE
1	boolean	hard	General Knowledge	The carbon cycle disrupts the balance of nitrogen in Earth's atmosphere, oceans, and biosphere	FALSE
1	boolean	hard	General Knowledge	The three types of blood vessels in the human body are arteries, veins, and capillaries	TRUE
1	boolean	hard	General Knowledge	Human digestion begins in the hand and ends in the large intestine	FALSE
1	boolean	hard	General Knowledge	The human digestive system breaks down food into nutrients.	TRUE
1	boolean	easy	General Knowledge	Adolf Hitler was born in Australia.	FALSE
1	boolean	easy	General Knowledge	The Sahara is the largest hot desert	TRUE

1	boolean	easy	General Knowledge	The sky is blue.	TRUE
1	boolean	easy	General Knowledge	The Mona Lisa is a famous painting by Leonardo da Vinci.	TRUE
1	boolean	easy	General Knowledge	Cars need soap to run.	FALSE
1	boolean	easy	General Knowledge	The greenhouse effect influences Earth's temperature.	TRUE
1	boolean	easy	General Knowledge	Apples are a type of fruit.	TRUE
1	boolean	easy	General Knowledge	Humans have five basic senses.	TRUE
1	boolean	easy	General Knowledge	The shortest river in the world is the Amazon River.	FALSE
1	boolean	easy	General Knowledge	Fossils destroy evidence of past life on Earth.	FALSE
1	boolean	hard	General Knowledge	Conduction is the transfer of heat through the stagnation of fluids or gases	FALSE
1	boolean	hard	General Knowledge	The Doppler effect prevents the change in frequency or wavelength of a wave in relation to an observer	FALSE
1	boolean	hard	General Knowledge	The process by which a solid turns directly into a gas is called sublimation	TRUE
1	boolean	hard	General Knowledge	The Krebs cycle is a series of chemical reactions that generate energy in cells.	TRUE
1	boolean	hard	General Knowledge	Mars has a thin atmosphere.	TRUE
1	boolean	hard	General Knowledge	Saturn's largest moon is Titan.	TRUE
1	boolean	hard	General Knowledge	Superconductors are materials that have infinite electrical resistance when cooled to certain temperatures.	FALSE
1	boolean	hard	General Knowledge	Deposition is the rapid building up of Earth's surface by natural processes	FALSE
1	boolean	hard	General Knowledge	Chemical reactions involve the conservation of atoms to maintain old substances.	FALSE
1	boolean	hard	General Knowledge	The water cycle includes evaporation, convection, precipitation, and collection.	TRUE
1	boolean	easy	General Knowledge	Humans do not use their brains.	FALSE
1	boolean	easy	General Knowledge	The coldest place on Earth is the equator.	FALSE
1	boolean	easy	General Knowledge	There are no planets in our solar system.	FALSE
1	boolean	easy	General Knowledge	Birds are not animals	FALSE
1	boolean	easy	General Knowledge	Water is poisonous to humans.	FALSE
1	boolean	easy	General Knowledge	Cows are mammals that produce milk.	TRUE
1	boolean	easy	General Knowledge	The Earth is located in the Milky Way galaxy.	TRUE
1	boolean	easy	General Knowledge	The sky is often cloudy when it's going to rain.	TRUE

1	boolean	easy	General Knowledge	Mount Everest is the shortest mountain in the world.	FALSE
1	boolean	easy	General Knowledge	The Nile River is located in South America.	FALSE
1	boolean	hard	General Knowledge	The atomic number of an element represents the number of electrons in its nucleus.	FALSE
1	boolean	hard	General Knowledge	Osmosis is the prevention of water movement across a selectively permeable membrane.	FALSE
1	boolean	hard	General Knowledge	Stars appear steady due to Earth's atmosphere.	FALSE
1	boolean	hard	General Knowledge	Polar ice caps are primarily made of fresh water.	TRUE
1	boolean	hard	General Knowledge	The planet Pluto has five known moons.	TRUE
1	boolean	hard	General Knowledge	The tallest tree in the world is a redwood tree named Hyperion.	TRUE
1	boolean	hard	General Knowledge	The four fundamental forces of nature are gravity, electromagnetism, the strong nuclear force, and the weak nuclear force.	TRUE
1	boolean	hard	General Knowledge	The planet Saturn is named after the Roman god of agriculture.	TRUE
1	boolean	hard	General Knowledge	The freezing point of water decreases as altitude increases	FALSE
1	boolean	hard	General Knowledge	The first successful powered flight was made by the Wright Brothers in 1903.	TRUE
1	boolean	easy	General Knowledge	Snow is cold	TRUE
1	boolean	easy	General Knowledge	Penguins can fly	FALSE
1	boolean	easy	General Knowledge	All animals are colorblind	FALSE
1	boolean	easy	General Knowledge	Earth is 71% land.	FALSE
1	boolean	easy	General Knowledge	The earth is round	TRUE
1	boolean	easy	General Knowledge	Dogs are not mammals	FALSE
1	boolean	easy	General Knowledge	Birds can fly	TRUE
1	boolean	easy	General Knowledge	The human body has bones	TRUE
1	boolean	easy	General Knowledge	A circle has 200 degrees	FALSE
1	boolean	easy	General Knowledge	Vaccines promote infectious diseases.	FALSE
1	boolean	easy	General Knowledge	Cats can bark like dogs	FALSE
1	boolean	easy	General Knowledge	Choclate is a popular dessert.	TRUE
1	boolean	easy	General Knowledge	Earth has a magnetic field	TRUE
1	boolean	easy	General Knowledge	Honey is produced by bees.	TRUE

1	boolean	easy	General Knowledge	Gravity makes things fall down	TRUE
1	boolean	hard	General Knowledge	Electromagnetic induction is the process by which a constant magnetic field dampens an electric current	FALSE
1	boolean	hard	General Knowledge	The planet Venus is often referred to as the "morning star" or the "evening star."	TRUE
1	boolean	hard	General Knowledge	The two main types of cells are prokaryotic and eukaryotic	TRUE
1	boolean	hard	General Knowledge	Our solar system consists of eight stars: Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, and Neptune	FALSE
1	boolean	hard	General Knowledge	A substance that can be broken down into simpler substances by chemical means is called an element.	FALSE
1	boolean	hard	General Knowledge	Water freezes at 0 degrees Celsius (32 °F) and boils at 100 degrees Celsius (212 °F)	TRUE
1	boolean	hard	General Knowledge	The process by which a gas turns directly into a solid, without becoming a liquid, is called sublimation	FALSE
1	boolean	hard	General Knowledge	Metamorphosis is a biological process in which an organism undergoes a significant change in form during its life cycle	TRUE
1	boolean	hard	General Knowledge	The auroras, or polar lights, are natural light displays caused by the interaction of solar particles with Earth's magnetic field	TRUE
1	boolean	hard	General Knowledge	The first law of thermodynamics states that energy cannot be created or destroyed, only converted from one form to another	TRUE
1	boolean	hard	General Knowledge	The planet Mars is known as the "Red Planet" due to its iron oxide-rich surface	TRUE
1	boolean	hard	General Knowledge	Radioactive decay occurs when stable atomic nuclei transform into more stable forms by emitting particles or radiation	FALSE
1	boolean	hard	General Knowledge	The process by which plants release carbon dioxide and absorb oxygen is called photosynthesis	FALSE
1	boolean	hard	General Knowledge	Sound waves require a medium to travel, such as air, water, or solids	TRUE
1	boolean	hard	General Knowledge	Black holes are regions of space where gravity is so strong that nothing, not even light, can escape	TRUE
test	boolean		General Knowledge	Apples grow on vines.	FALSE
test	boolean		General Knowledge	The smallest animal in the world is the elephant.	FALSE
test	boolean		General Knowledge	Comets are icy celestial objects.	TRUE
test	boolean		General Knowledge	The study of the universe beyond Earth's atmosphere is called astronomy.	TRUE
test	boolean		General Knowledge	The fastest bird in the world is the penguin.	FALSE
test	boolean		General Knowledge	Mars has two small moons, Phobos and Deimos.	TRUE

Appendix 6.

Post questionnaire about self-reporting tendencies.

Bachelor-Thesis - complete randomization

Start of Block: Introduction_and_Consent

[Briefing]
Dear Participant,   Welcome to this study. The purpose of this research is to investigate robotic gaze behavior and joint attention in human-robot interaction and collaboration. In this study, you play a simple classification game where you have to sort cards into one out of two categories while interacting with a screen-based robot.   **Study Design:** In this repeated-measures study, you will be randomly assigned to robots that display different gaze skills. In interaction with every robot, you will classify statements from a stack of cards into True or False categories. In total, you will answer 90 statements. After the experimental game, you have to answer a survey regarding your experience in the game. **Support:** Please note that you can withdraw from this study at any point. If you feel the need to talk to someone about the presented information, do not hesitate to call the following number. The Netherlands: 0800 0113.   **Confidentiality:** We understand that the information you provide is sensitive. Thus, we want to ensure that all your data will be kept confidential. Any data or other information that could directly identify you will be removed from your responses before analysis. All data collected during this study will be stored securely. Data access will be provided only to the research team of this study.   **Anonymisation:** Your name and any other information that could directly identify you will be removed from your responses before analysis. We will assign you a unique code number to track your data throughout the study.   **Secure Storage:** All data collected during this study will be stored securely on a password-protected computer of the researcher. Only the research team will have access to this data.   **Reporting:** Any reports or publications resulting from this study will not include any information that could identify you.   **Contact Information and Right to Withdraw:** In case you have any further questions about the study, or if you want to withdraw from the study after you have consented, you can always contact one of the researchers at the following E-mail address. You can also contact the University of Twente Psychology Department Ethics Committee at ethicscommittee-hss@utwente.nl if you have any concerns about how the study is being conducted.   **Researcher:** Devin Kruse (d.kruse-1@student.utwente.nl) +49 162 337 2000

[Consent]
Please tick the appropriate boxes

[understandingStudy]
I have read (or it has been read to me) and understood the study information. I have been able to ask questions about the study and my questions have been answered to my satisfaction.

○ Yes (1)

○ No (2)

[voluntaryConsent]
I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.

○ Yes (1)

○ No (2)

[understandingDesign]
I understand that taking part in the study involves the interaction with a screen-based, real-time animated robot in an experimental game.

○ Yes (1)

○ No (2)

[publication]
I understand that information I provide will be used for publication on scientific databases.

- Yes (1)
- No (2)

[dataStoring]
I give permission for the unpersonalised questionnaire data that I provide to be archived in the database of the University of Twente so it can be used for future research and learning.

- Yes (1)
- No (2)

End of Block: Introduction_and_Consent

Start of Block: Demographics

[Age]
What is your date of birth? "Please enter in the format: DD-MM-YYYY"

____

[residence]
Country of Residence:

- The Netherlands (1)
- Germany (2)
- Other (3) ____

[gender]
What is your Gender Identity?

- Male (1)
- Female (2)
- Non-binary / third gender (3)
- Prefer not to say (4)

[student]
Are you currently a student in a University/College?

◯ Yes (1)

◯ No (2)

[participantID]
What is your Participant-ID?

____

End of Block: Demographics

Start of Block: Game Instruction

[explanation]
Experimental Set-Up: In the game, you will see the TRUE category in the box on the left and the FALSE category in the box on the right. These categories remain the same during the entire experiment. In the lower half of the screen, you'll see a stack of cards. Once you tap one of the cards, it will be revealed and can then be pushed to one of the sides. For each card reveal, you will see one of three different robots. The robots display different skills and behaviors. After you have dropped the statement into one of the categories, you will see a different robot. Note that you will not see the same robot for two questions in a row. In total, you have to categorize 90 statements for this game. Before you start the game, you can do a test round with 6 statements. For testing and the very first card, you will not see the robot but a black box which displays the text "ready?!" **How the game works**: As mentioned above, you can see two categories, TRUE and FALSE, on your left and right sides. In the game, your task is to classify the revealing cards. A card is revealed when you touch on the stack of cards. When you consider the statement to be true, move it to the left category. When you consider the statement to be false, move it to the right category. Before you move a card, try to build eye contact with the robot and be attentive to the robots behavioral cues. **Important:** There's no time limit on the game. However, you should try to categorize each statement as quickly as possible, so do not overthink too much and try to listen to your intuition. For us, it is more important to see how you interact with the robot and maintain eye contact during the game rather than how you scored in the game. Try to **create eye-contact with the robot** after each card reveal. Note: **The robots beliefs can be based on a random belief model**, so you have to **decide whether you trust the robot or not**.

End of Block: Game Instruction

Start of Block: Robot_Introduction

[RobotIntroduction]
As mentioned above, you will play the game together with 3 different robots, that have different skills. The robots that you will meet in the game are Carl, Ryan and Ivan. Take a moment and have a look on the pictures to get familiar with them. After that, go to the next page. The robots can look a little bit similar, so try to remember some of the differences from these pictures. During the game, you will also see the name of each robot in the top left corner. **Your goal is to interact with each of the robots during the game. This is Carl: This is Ryan: And this is Ivan:**

End of Block: Robot_Introduction

Start of Block: ready1

You can start the experiment now. Please tell the researcher that you are ready before you continue with this questionnaire!

End of Block: ready1

Start of Block: finishGame

Thanks for your participation in the experimental game. In the following minutes, we will ask you for your perception on the 3 robots. There is no right or wrong. Simply share your own intuition and perception of the robots regarding the questions and statements. Please go to the next page when you are ready.

End of Block: finishGame

Start of Block: Carl_evaluation

Lets start with **Carl**: Please rate your overall impression of Carl based on the following descriptive words. For each pair of words, please indicate where you feel Carl falls on the spectrum between them.

[anthropomorphism]
How did you perceive Carl on the following spectra:

	1 (1)	2 (2)	3 (3)	4 (4)	5 (5)	
Fake	○	○	○	○	○	Natural
Machinelike	○	○	○	○	○	Humanlike
Unconscious	○	○	○	○	○	Conscious
Artificial	○	○	○	○	○	Lifelike
Moving rigidly	○	○	○	○	○	Moving elegantly

[likability]
How did you perceive Carl on the following spectra:

	1 (1)	2 (2)	3 (3)	4 (4)	5 (5)	
Dislike	○	○	○	○	○	Like
Unfriendly	○	○	○	○	○	Friendly
Unkind	○	○	○	○	○	Kind
Unpleasant	○	○	○	○	○	Pleasant
Awful	○	○	○	○	○	Nice

[intelligence]
How did you perceive Carl on the following spectra:

	1 (1)	2 (2)	3 (3)	4 (4)	5 (5)	
Incompetent	○	○	○	○	○	Competent
Ignorant	○	○	○	○	○	Knowledgeable
Irresponsive	○	○	○	○	○	Responsible
Unintelligent	○	○	○	○	○	Intelligent
Foolish	○	○	○	○	○	Sensible

[trust]
Please rate your agreement with the following statements about Carl:

	Strongly Disagree (1)	Disagree (2)	Somewhat Disagree (3)	Neutral (4)	Somewhat Agree (5)	Agree (6)	Strongly Agree (7)
The robot functions successfully. (1)	○	○	○	○	○	○	○
The robot acts consistently (2)	○	○	○	○	○	○	○
The robot is reliable (3)	○	○	○	○	○	○	○
The robot is predictable. (4)	○	○	○	○	○	○	○
The robot is dependable. (5)	○	○	○	○	○	○	○
The robot follows directions. (6)	○	○	○	○	○	○	○
The robot meets the needs of the mission. (7)	○	○	○	○	○	○	○
The robot performs exactly as instructed. (8)	○	○	○	○	○	○	○
The robot has errors (9)	○	○	○	○	○	○	○
The robot provides appropriate information. (10)	○	○	○	○	○	○	○
The robot malfunctions. (11)	○	○	○	○	○	○	○
The robot communicates with people. (12)	○	○	○	○	○	○	○
The robot provides feedback. (13)	○	○	○	○	○	○	○
The robot is unresponsive. (14)	○	○	○	○	○	○	○

End of Block: Carl_evaluation

Start of Block: Ryan_evaluation

Lets continue with **Ryan:**Please rate your overall impression of Ryan based on the following descriptive words. For each pair of words, please indicate where you feel Ryan falls on the spectrum between them.

[anthropomorphism]
How did you perceive Ryan on the following spectra:

	1 (1)	2 (2)	3 (3)	4 (4)	5 (5)	
Fake	○	○	○	○	○	Natural
Machinelike	○	○	○	○	○	Humanlike
Unconscious	○	○	○	○	○	Conscious
Artificial	○	○	○	○	○	Lifelike
Moving rigidly	○	○	○	○	○	Moving elegantly

[likability]
How did you perceive Ryan on the following spectra:

	1 (1)	2 (2)	3 (3)	4 (4)	5 (5)	
Dislike	○	○	○	○	○	Like
Unfriendly	○	○	○	○	○	Friendly
Unkind	○	○	○	○	○	Kind
Unpleasant	○	○	○	○	○	Pleasant
Awful	○	○	○	○	○	Nice

[intelligence]
How did you perceive Ryan on the following spectra:

	1 (1)	2 (2)	3 (3)	4 (4)	5 (5)	
Incompetent	○	○	○	○	○	Competent
Ignorant	○	○	○	○	○	Knowledgeable
Irresponsive	○	○	○	○	○	Responsible
Unintelligent	○	○	○	○	○	Intelligent
Foolish	○	○	○	○	○	Sensible

[trust]
Please rate your agreement with the following statements about Ryan:

	Strongly Disagree (1)	Disagree (2)	Somewhat Disagree (3)	Neutral (4)	Somewhat Agree (5)	Agree (6)	Strongly Agree (7)
The robot functions successfully. (1)	◯	◯	◯	◯	◯	◯	◯
The robot acts consistently (2)	◯	◯	◯	◯	◯	◯	◯
The robot is reliable (3)	◯	◯	◯	◯	◯	◯	◯
The robot is predictable. (4)	◯	◯	◯	◯	◯	◯	◯
The robot is dependable. (5)	◯	◯	◯	◯	◯	◯	◯
The robot follows directions. (6)	◯	◯	◯	◯	◯	◯	◯
The robot meets the needs of the mission. (7)	◯	◯	◯	◯	◯	◯	◯
The robot performs exactly as instructed. (8)	◯	◯	◯	◯	◯	◯	◯
The robot has errors (9)	◯	◯	◯	◯	◯	◯	◯
The robot provides appropriate information. (10)	◯	◯	◯	◯	◯	◯	◯
The robot malfunctions. (11)	◯	◯	◯	◯	◯	◯	◯
The robot communicates with people. (12)	◯	◯	◯	◯	◯	◯	◯
The robot provides feedback. (13)	◯	◯	◯	◯	◯	◯	◯
The robot is unresponsive. (14)	◯	◯	◯	◯	◯	◯	◯

End of Block: Ryan_evaluation

Start of Block: Ivan_evaluation

Lets continue with **Ivan:** Please rate your overall impression of Ivan based on the following descriptive words. For each pair of words, please indicate where you feel Ivan falls on the spectrum between them.

[anthropomorphism]
How did you perceive Ivan on the following spectra:

	1 (1)	2 (2)	3 (3)	4 (4)	5 (5)	
Fake	○	○	○	○	○	Natural
Machinelike	○	○	○	○	○	Humanlike
Unconscious	○	○	○	○	○	Conscious
Artificial	○	○	○	○	○	Lifelike
Moving rigidly	○	○	○	○	○	Moving elegantly

[likability]
How did you perceive Ivan on the following spectra:

	1 (1)	2 (2)	3 (3)	4 (4)	5 (5)	
Dislike	○	○	○	○	○	Like
Unfriendly	○	○	○	○	○	Friendly
Unkind	○	○	○	○	○	Kind
Unpleasant	○	○	○	○	○	Pleasant
Awful	○	○	○	○	○	Nice

[intelligence]
How did you perceive Ivan on the following spectra:

	1 (1)	2 (2)	3 (3)	4 (4)	5 (5)	
Incompetent	○	○	○	○	○	Competent
Ignorant	○	○	○	○	○	Knowledgeable
Irresponsive	○	○	○	○	○	Responsible
Unintelligent	○	○	○	○	○	Intelligent
Foolish	○	○	○	○	○	Sensible

[trust]
Please rate your agreement with the following statements about Ivan:

	Strongly Disagree (1)	Disagree (2)	Somewhat Disagree (3)	Neutral (4)	Somewhat Agree (5)	Agree (6)	Strongly Agree (7)
The robot functions successfully. (1)	○	○	○	○	○	○	○
The robot acts consistently (2)	○	○	○	○	○	○	○
The robot is reliable (3)	○	○	○	○	○	○	○
The robot is predictable. (4)	○	○	○	○	○	○	○
The robot is dependable. (5)	○	○	○	○	○	○	○
The robot follows directions. (6)	○	○	○	○	○	○	○
The robot meets the needs of the mission. (7)	○	○	○	○	○	○	○
The robot performs exactly as instructed. (8)	○	○	○	○	○	○	○
The robot has errors (9)	○	○	○	○	○	○	○
The robot provides appropriate information. (10)	○	○	○	○	○	○	○
The robot malfunctions. (11)	○	○	○	○	○	○	○
The robot communicates with people. (12)	○	○	○	○	○	○	○
The robot provides feedback. (13)	○	○	○	○	○	○	○
The robot is unresponsive. (14)	○	○	○	○	○	○	○

End of Block: Ivan_evaluation

Appendix 7.

Included Data Analysis Scripts.

Note. The Data Analysis Scripts should be also find online using the links or barcodes provided in Appendix 3.

Python Script for Data Cleaning and Transformation for the Post-Questionnaire Qualtrics Data

```
import pandas as pd
import numpy as np
import re

# --- Configuration ---
# Input file from the original first script
INPUT_CSV_FILE = 'qualtrics_questionnaire.csv'
# Final output file from the original second script
OUTPUT_CSV_FILE = 'qualtrics_data_final.csv'

# --- Part 1: Configuration from the first script ---
# Define the initial columns you absolutely want to keep by their base name
INITIAL_COLS_TO_KEEP = [
    'understandingStudy', 'voluntaryConsent', 'understandingDesign',
    'publication', 'dataStoring', 'Age', 'residence', 'gender',
    'student', 'participantID'
]

# Likert scale mapping for TRUST variables
LIKERT_MAPPING_TRUST = {
    "Strongly Disagree": 1,
    "Disagree": 2,
    "Somewhat Disagree": 3,
    "Neutral": 4,
    "Somewhat Agree": 5,
    "Agree": 6,
    "Strongly Agree": 7
}

# Base names of TRUST items to be reverse-coded
TRUST_ITEMS_TO_REVERSE = ['trust_9', 'trust_11', 'trust_14']

# --- Part 2: Configuration from the second script ---
# Manual override for column types during imputation.
# Set to a list of column names to override auto-detection, otherwise leave as None.
MANUAL_NUMERIC_COLS = None
```

```
MANUAL_CATEGORICAL_COLS = None

# --- Helper Functions (from Script 1) ---
def clean_column_name(col_name):
    """Cleans a column name by stripping whitespace and replacing non-breaking spaces."""
    if pd.isna(col_name):
        return f"Unnamed_Column_{pd.Timestamp.now().nanosecond}"
    return str(col_name).strip().replace('\xa0', ' ')

def generate_new_column_names(original_headers, metadata_row_values):
    """Generates new, unique column names based on metadata."""
    cleaned_original_headers = [clean_column_name(h) for h in original_headers]
    pid_cleaned_name = clean_column_name('participantID')

    try:
        pid_idx = cleaned_original_headers.index(pid_cleaned_name)
    except ValueError:
        print(f"CRITICAL ERROR: '{pid_cleaned_name}' column not found in the CSV headers.
Cannot proceed.")
        return None

    new_names = []
    for i, name in enumerate(cleaned_original_headers):
        new_name = name
        if i > pid_idx:
            metadata_str = str(metadata_row_values[i]).lower()
            suffix = ""
            if "carl" in metadata_str: suffix = "_carl"
            elif "ryan" in metadata_str: suffix = "_ryan"
            elif "ivan" in metadata_str: suffix = "_ivan"
            if suffix: new_name = name + suffix
        new_names.append(new_name)

    # Ensure final names are unique by appending .1, .2, etc. if needed
    final_unique_names = []
    counts = {}
    for name in new_names:
        if name not in counts:
            counts[name] = 0
            final_unique_names.append(name)
        else:
            counts[name] += 1
            final_unique_names.append(f"{name}.{counts[name]}")

    return final_unique_names
```

```
# --- Processing Functions (from Script 1) ---
def convert_likert_scales(df, column_prefix, mapping):
    """Converts columns with Likert scale text to numeric values."""
    print(f"\n--- Converting Likert Scales for columns starting with '{column_prefix}' ---")
    converted_cols_count = 0
    for col in df.columns:
        if str(col).lower().startswith(column_prefix.lower()):
            print(f"  Converting column: {col}")
            df[col] = df[col].astype(str).map(mapping)
            df[col] = pd.to_numeric(df[col], errors='coerce')
            converted_cols_count += 1
            if df[col].isnull().any():
                print(f"    Note: Some values in '{col}' became NaN (could not be mapped or were
already NaN).")
    if converted_cols_count == 0:
        print(f"  No columns found starting with '{column_prefix}' for Likert conversion.")
    else:
        print(f"  Successfully attempted Likert conversion for {converted_cols_count} columns.")
    return df

def reverse_code_items(df, items_to_reverse_bases, scale_min=1, scale_max=7):
    """Reverse codes specified numeric columns based on a scale."""
    print("\n--- Reverse Coding Specific Trust Items ---")
    reverse_value = scale_min + scale_max
    reversed_cols_found_count = 0

    for col in df.columns:
        for base_item in items_to_reverse_bases:
            # Regex to match base_item, optional suffix, and optional duplicate number (.1)
            pattern = rf"^{re.escape(base_item)}(_carl|_ryan|_ivan)?(\.\d+)?$"
            if re.match(pattern, str(col).lower()):
                if pd.api.types.is_numeric_dtype(df[col]):
                    if df[col].notna().any():
                        print(f"  Reverse coding column: {col} (Original mean: {df[col].mean():.2f})")
                        df[col] = reverse_value - df[col]
                        print(f"    New mean for {col}: {df[col].mean():.2f}")
                    else:
                        print(f"  Column {col} contains all NaNs, skipping reverse coding logic.")
                    reversed_cols_found_count += 1
                else:
                    print(f"    Warning: Column '{col}' identified for reverse coding is not numeric.
Skipping.")
                break

    if reversed_cols_found_count == 0:
        print("  No columns found matching the criteria for reverse coding.")
```

```
    else:
        print(f"  Successfully attempted reverse coding for {reversed_cols_found_count}
columns.")
    return df

def reorder_columns(df, first_col_name_base):
    """Moves a specified column to the first position in the DataFrame."""
    print(f"\n--- Reordering Columns to make '{first_col_name_base}' first ---")
    target_first_col_final_name = None
    cleaned_first_col_base = clean_column_name(first_col_name_base)

    if cleaned_first_col_base in df.columns:
        target_first_col_final_name = cleaned_first_col_base
    else:
        for col_name_in_df in df.columns:
            if str(col_name_in_df).startswith(cleaned_first_col_base):
                target_first_col_final_name = col_name_in_df
                print(f"  Found '{first_col_name_base}' as column '{target_first_col_final_name}'.")
                break

    if target_first_col_final_name and target_first_col_final_name in df.columns:
        cols = [target_first_col_final_name] + [col for col in df.columns if col !=
target_first_col_final_name]
        df = df[cols]
        print(f"  Column '{target_first_col_final_name}' moved to the first position.")
    else:
        print(f"  Warning: Column based on '{first_col_name_base}' not found. No reordering
done.")
    return df

# --- Imputation Functions (from Script 2) ---
def report_missing_values(df, title="Missing Value Report"):
    """Prints a report of missing values (count and percentage) for each column."""
    print(f"\n--- {title} ---")
    missing_count = df.isnull().sum()
    missing_percentage = (missing_count / len(df)) * 100
    missing_df = pd.DataFrame({
        'Missing Count': missing_count,
        'Missing Percentage (%)': missing_percentage
    })
    missing_df = missing_df[missing_df['Missing Count'] > 0].sort_values(by='Missing
Percentage (%)', ascending=False)

    if missing_df.empty:
        print("No missing values found in the dataset.")
    else:
```

```
        print(missing_df)
    return missing_df

def impute_missing_data(df, numeric_cols_override=None, categorical_cols_override=None):
    """Imputes missing data: median for numeric, mode for categorical."""
    print("\n--- Starting Data Imputation ---")
    df_imputed = df.copy()

    # Determine numeric columns for imputation
    if numeric_cols_override is not None:
        numeric_cols = [col for col in numeric_cols_override if col in df_imputed.columns]
        print(f"Using manually specified numeric columns: {numeric_cols}")
    else:
        numeric_cols = df_imputed.select_dtypes(include=np.number).columns.tolist()
        print(f"Auto-detected numeric columns for imputation: {numeric_cols}")

    # Determine categorical/object columns for imputation
    if categorical_cols_override is not None:
        categorical_cols = [col for col in categorical_cols_override if col in df_imputed.columns]
        print(f"Using manually specified categorical columns: {categorical_cols}")
    else:
        all_cols = df_imputed.columns.tolist()
        categorical_cols = [col for col in all_cols if col not in numeric_cols]
        print(f"Auto-detecting categorical/object columns (all non-numeric): {categorical_cols}")

    # Impute numeric columns with MEDIAN
    for col in numeric_cols:
        if df_imputed[col].isnull().any():
            median_val = df_imputed[col].median()
            df_imputed[col].fillna(median_val, inplace=True)
            print(f"  Numeric column '{col}': Imputed NaNs with median ({median_val:.2f})")

    # Impute categorical/object columns with MODE
    for col in categorical_cols:
        if df_imputed[col].isnull().any():
            if df_imputed[col].dtype == 'object' or
pd.api.types.is_categorical_dtype(df_imputed[col]):
                mode_val = df_imputed[col].mode()
                if not mode_val.empty:
                    mode_val = mode_val[0]
                    df_imputed[col].fillna(mode_val, inplace=True)
                    print(f"  Categorical column '{col}': Imputed NaNs with mode ('{mode_val}')")
                else:
                    print(f"  Categorical column '{col}': Mode could not be determined. NaNs remain.")
            else:
                print(f"  Skipping imputation for '{col}' as it is not an object/category type.")
```

```
    print("--- Imputation Attempt Finished ---")
    return df_imputed

# --- Main Script Execution ---
def main():
    print(f"--- Starting Full Pipeline: Processing {INPUT_CSV_FILE} ---")

    #
==============================================================================
====
    # STAGE 1: DATA LOADING AND CLEANING (from Script 1)
    #
==============================================================================
====
    try:
        df_headers = pd.read_csv(INPUT_CSV_FILE, header=None, nrows=1, encoding='utf-8')
        original_headers = df_headers.iloc[0].tolist()
        df_metadata_row = pd.read_csv(INPUT_CSV_FILE, header=None, nrows=1,
skiprows=[0], encoding='utf-8')
        metadata_row_values = df_metadata_row.iloc[0].tolist()
    except FileNotFoundError:
        print(f"Error: Input file '{INPUT_CSV_FILE}' not found.")
        return
    except Exception as e:
        print(f"Error reading header/metadata rows: {e}")
        return

    final_column_names = generate_new_column_names(original_headers,
metadata_row_values)
    if final_column_names is None: return

    try:
        df_data = pd.read_csv(INPUT_CSV_FILE, header=None, skiprows=2,
names=final_column_names, encoding='utf-8', dtype=str, keep_default_na=False)
    except Exception as e:
        print(f"Error reading main data: {e}")
        return

    df_data.replace('', np.nan, inplace=True)
    print(f"\nDataFrame loaded with {df_data.shape[0]} data rows and {df_data.shape[1]}
columns.")

    # --- Column Selection ---
    columns_to_keep_final = []
    initial_cols_cleaned = [clean_column_name(col) for col in INITIAL_COLS_TO_KEEP]
```

```
    original_headers_cleaned_for_selection = [clean_column_name(h) for h in original_headers]

    for initial_col_name_to_find in initial_cols_cleaned:
        found_in_original = False
        for i, original_cleaned_h in enumerate(original_headers_cleaned_for_selection):
            if original_cleaned_h == initial_col_name_to_find:
                if final_column_names[i] not in columns_to_keep_final:
                    columns_to_keep_final.append(final_column_names[i])
                found_in_original = True
                break
        if not found_in_original:
            print(f"Warning during selection: Initial column '{initial_col_name_to_find}' was not
found.")

    pid_original_cleaned_name = clean_column_name('participantID')
    try:
        pid_original_idx =
original_headers_cleaned_for_selection.index(pid_original_cleaned_name)
        for i in range(pid_original_idx, len(final_column_names)):
            if final_column_names[i] not in columns_to_keep_final:
                 columns_to_keep_final.append(final_column_names[i])
    except ValueError:
        print(f"CRITICAL ERROR: Original '{pid_original_cleaned_name}' column not found.")
        return

    print(f"\nColumns selected (count: {len(columns_to_keep_final)}):
{str(columns_to_keep_final[:10])[:200]}...")

    try:
        df_selected = df_data[columns_to_keep_final].copy()
    except KeyError as e:
        print(f"KeyError during final column selection: {e}.")
        return

    print(f"Shape after column selection: {df_selected.shape}")

    # --- Data Processing ---
    df_processed = convert_likert_scales(df_selected, "trust", LIKERT_MAPPING_TRUST)
    df_processed = reverse_code_items(df_processed, TRUST_ITEMS_TO_REVERSE)
    df_processed_reordered = reorder_columns(df_processed, 'participantID')

    # --- Remove first data row ---
    if not df_processed_reordered.empty:
        print("\n--- Removing the first data row (metadata/test row) ---")
        df_processed_final = df_processed_reordered.iloc[1:].reset_index(drop=True)
        print(f"Shape after removing first data row: {df_processed_final.shape}")
```

```
    else:
        print("\nDataFrame is empty prior to removal of the first data row.")
        df_processed_final = df_processed_reordered

    #
==============================================================================
====
    # STAGE 2: DATA IMPUTATION (from Script 2)
    #
==============================================================================
====

    # Initial missing value report on the processed data
    report_missing_values(df_processed_final, title="Missing Value Report (Before Imputation)")

    # Perform imputation
    df_imputed = impute_missing_data(df_processed_final,
                                  numeric_cols_override=MANUAL_NUMERIC_COLS,
                                  categorical_cols_override=MANUAL_CATEGORICAL_COLS)

    # Final missing value report after imputation
    report_missing_values(df_imputed, title="Missing Value Report (After Imputation)")

    #
==============================================================================
====
    # STAGE 3: SAVE FINAL OUTPUT
    #
==============================================================================
====
    try:
        df_imputed.to_csv(OUTPUT_CSV_FILE, index=False, encoding='utf-8')
        print(f"\nSuccessfully saved final data to '{OUTPUT_CSV_FILE}'.")
        print(f"Final shape of saved data: {df_imputed.shape}")
        print(f"Final columns (first 10): {list(df_imputed.columns)[:10]}")
        print("\nFirst 5 rows of the final imputed data:")
        print(df_imputed.head())

        # Final info and stats
        print("\nInfo for final data:")
        df_imputed.info()
        trust_cols_final = [col for col in df_imputed.columns if str(col).lower().startswith('trust')
and pd.api.types.is_numeric_dtype(df_imputed[col])]
        if trust_cols_final:
            print("\nDescriptive statistics for numeric 'trust' columns in the final data:")
            print(df_imputed[trust_cols_final].describe())
```

```
    except Exception as e:
        print(f"\nError saving final imputed data to CSV: {e}")

if __name__ == '__main__':
    main()
```

Python Script for Data Cleaning and Transformation from the Performance and Gaze Following Data

```
# Import necessary libraries
import os
import glob
import pandas as pd

# Define the path to the folder containing the gaze log files
# Assumes the 'gaze_logs' folder is in the same directory as the script
folder_path = 'gaze_logs'

# Define the pattern for the gaze log files
# It looks for files starting with 'gaze_log_p' and ending with '.csv'
file_pattern = os.path.join(folder_path, 'gaze_log_p*.csv')

# Find all files in the folder that match the pattern
all_files = glob.glob(file_pattern)

# Check if any files were found
if not all_files:
    print(f"No files matching the pattern '{file_pattern}' found in the folder '{folder_path}'.")
else:
    print(f"Found {len(all_files)} files to combine:")
    for f in all_files:
        print(f" - {os.path.basename(f)}")

    # Initialize an empty list to hold DataFrames
    list_of_dfs = []

    # Loop through the list of files found
    for filename in all_files:
        try:
            # Read the current CSV file into a DataFrame
            df = pd.read_csv(filename, index_col=None, header=0)
            # Add the DataFrame to the list
            list_of_dfs.append(df)
            print(f"Successfully read {os.path.basename(filename)}")
        except Exception as e:
            print(f"Error reading {os.path.basename(filename)}: {e}")

    # Check if any DataFrames were successfully read
    if not list_of_dfs:
        print("No dataframes were created. Cannot proceed.")
    else:
        # Concatenate all DataFrames in the list into a single DataFrame
```

```
    combined_df = pd.concat(list_of_dfs, axis=0, ignore_index=True)

    # Define the name for the output file
    output_filename = 'total_gaze.csv'

    # Save the combined DataFrame to a new CSV file
    try:
        combined_df.to_csv(output_filename, index=False)
        print(f"\nSuccessfully combined {len(list_of_dfs)} files into '{output_filename}'.")
        print(f"The combined file has {combined_df.shape[0]} rows and
{combined_df.shape[1]} columns.")
    except Exception as e:
        print(f"Error writing the combined file '{output_filename}': {e}")
```

Python Script for Data Cleaning and Data Transformation of the Eye-tracking Data

```
#!/usr/bin/env python3
import pandas as pd
import os
import re
from pathlib import Path

# --- MASTER CONFIGURATION ---
# === Inputs ===
# Directory for original eye-tracking TSV files (from script 1)
EYETRACKING_INPUT_DIR = Path('eyetracking_files')
# Directory for gaze log CSV files (from script 2)
GAZE_LOG_DIR = Path('gaze_files')

# === Output ===
# Final combined CSV file name (from script 3)
FINAL_OUTPUT_CSV = "combined_eyetracking_data.csv"

# === Processing Parameters (from scripts 1 & 2) ===
# Main AOI categories to look for in eye-tracking data
AOI_CATEGORIES = [
    'cards', 'eyes', 'face', 'false_category',
    'robot', 'robot_name', 'true_category'
]

# Columns needed for timestamp calculation
TIMESTAMP_COLUMNS = ['Recording date UTC', 'Recording start time UTC', 'Recording
timestamp']

# Columns from gaze_log files essential for processing
BASE_REQUIRED_GAZE_COLS = ['timestamp', 'move_duration', 'participant']

# Additional columns from gaze_log to merge into the final output
GAZE_COLS_TO_MERGE = [
    'difficulty', 'correct_answer', 'correct_side',
    'participants_side_choice', 'Robot', 'gazeDecision'
]

# Define the EXACT final columns for the output file
FINAL_OUTPUT_COLUMNS = [
    'Eyetracker timestamp', 'Gaze point X (MCSnorm)', 'Gaze point Y (MCSnorm)',
    'Pupil diameter left', 'Pupil diameter right', 'Validity left', 'Validity right',
    'Eye movement type', 'Eye movement event duration',
    'Fixation point X (MCSnorm)', 'Fixation point Y (MCSnorm)',
    'Event', 'Event value', 'Mouse position X', 'Mouse position Y',
```

```
    'ts_utc',
    'is_cards', 'is_eyes', 'is_face', 'is_false_category', 'is_robot',
    'is_robot_name', 'is_true_category', 'active_areas',
    'ParticipantID',
    'classification_timeframe_number',
    'robot_appearance_timeframe_number'
] + GAZE_COLS_TO_MERGE


# --- HELPER & PROCESSING FUNCTIONS (Combined from all scripts) ---

def get_participant_id_from_filename(filename_str):
    """Extracts participant ID (e.g., 'p1') from a filename."""
    match = re.search(r'_p(\d+)', filename_str)
    if match:
        return f"p{match.group(1)}"
    match_direct = re.match(r'p(\d+)', Path(filename_str).stem.split('_')[-1])
    if match_direct:
        return f"p{match_direct.group(1)}"
    return Path(filename_str).stem

def add_aoi_columns(df, aoi_categories_list):
    """
    (From Script 1) Adds boolean AOI and 'active_areas' columns to a DataFrame in memory.
    """
    print("  Step 1a: Processing AOI categories to create boolean flags...")
    for category in aoi_categories_list:
        # Regex to find the column for a specific AOI category
        regex_pattern = re.compile(f"AOI hit \\[Web Page Recording.*? - {re.escape(category)}\\]",
re.IGNORECASE)

        potential_aoi_columns = [col for col in df.columns if regex_pattern.fullmatch(col)]
        selected_column_for_category = None

        if potential_aoi_columns:
            for col_name in potential_aoi_columns:
                # Check if the column has any non-null, non-zero data
                if df[col_name].notna().any():
                    series_numeric = pd.to_numeric(df[col_name], errors='coerce')
                    if series_numeric[series_numeric.notna()].astype(bool).any():
                        selected_column_for_category = col_name
                        break

        bool_col_name = f"is_{category}"
        if selected_column_for_category:
            numeric_series = pd.to_numeric(df[selected_column_for_category], errors='coerce')
```

```
            df[bool_col_name] = numeric_series.notna() & numeric_series.astype(bool)
            print(f"    - Created boolean column '{bool_col_name}' from
'{selected_column_for_category}'.")
        else:
            df[bool_col_name] = False
            print(f"    - No active AOI column found for '{category}'. '{bool_col_name}' set to
False.")

    # Create 'active_areas' column
    def determine_active_areas(row):
        active_names = [cat for cat in aoi_categories_list if row.get(f"is_{cat}", False)]
        return ", ".join(active_names) if active_names else pd.NA

    df['active_areas'] = df.apply(determine_active_areas, axis=1)
    print("    - Generated 'active_areas' column.")
    return df

def process_single_participant(participant_id, raw_tobii_filepath, gaze_log_filepath):
    """
    (Combines logic from scripts 1 & 2)
    Processes a single participant's data from raw files to a final, merged DataFrame.
    Returns a DataFrame for one participant, or None if an error occurs.
    """
    print(f"\n--- Processing Participant: {participant_id} ---")

    # 1. LOAD GAZE LOG DATA (from Script 2)
    print(f"  Loading gaze log: {gaze_log_filepath.name}")
    try:
        gaze_df = pd.read_csv(gaze_log_filepath)
        # Check for essential columns
        if any(col not in gaze_df.columns for col in BASE_REQUIRED_GAZE_COLS):
            print(f"  Error: Gaze log is missing one of required columns:
{BASE_REQUIRED_GAZE_COLS}. Skipping.")
            return None
        participant_id_col = int(gaze_df['participant'].dropna().iloc[0])
    except Exception as e:
        print(f"  Error reading or parsing gaze log file {gaze_log_filepath.name}: {e}. Skipping.")
        return None

    # Ensure all columns to be merged exist, adding them as NA if not
    for col in GAZE_COLS_TO_MERGE:
        if col not in gaze_df.columns:
            gaze_df[col] = pd.NA

    # 2. DEFINE TIMEFRAMES FROM GAZE LOG (from Script 2)
    print("  Step 1b: Defining timeframes from gaze log...")
```

```
    try:
        gaze_df['ts_utc'] = pd.to_datetime(gaze_df['timestamp'], utc=True, errors='coerce')
        gaze_df['move_duration'] = pd.to_numeric(gaze_df['move_duration'], errors='coerce')
        gaze_df.dropna(subset=['ts_utc', 'move_duration'], inplace=True)
        gaze_df = gaze_df.sort_values('ts_utc').reset_index(drop=True)

        gaze_df['classification_time_start'] = gaze_df['ts_utc']
        gaze_df['classification_time_end'] = gaze_df['ts_utc'] +
pd.to_timedelta(gaze_df['move_duration'], unit='s')
        gaze_df['classification_timeframe_number_val'] = range(1, len(gaze_df) + 1)

        gaze_df['robot_appearance_time_start'] = gaze_df['classification_time_end'].shift(1)
        gaze_df['robot_appearance_time_end'] = gaze_df['classification_time_end']
        gaze_df['robot_appearance_timeframe_number_val'] = range(1, len(gaze_df) + 1)

        gaze_df.dropna(subset=['classification_time_start', 'classification_time_end',
'robot_appearance_time_end'], inplace=True)
        if gaze_df.empty:
            print("  Error: No valid timeframes could be defined from gaze log. Skipping.")
            return None

        overall_start = gaze_df['classification_time_start'].min()
        overall_end = gaze_df['classification_time_end'].max()
    except Exception as e:
        print(f"  Error defining timeframes for {participant_id}: {e}. Skipping.")
        return None

    # 3. LOAD & PROCESS RAW EYE-TRACKING DATA (combining scripts 1 & 2)
    print(f"  Loading raw eye-tracking data: {raw_tobii_filepath.name}")
    try:
        et_df = pd.read_csv(raw_tobii_filepath, sep='\t', low_memory=False)
        if any(col not in et_df.columns for col in TIMESTAMP_COLUMNS):
            print(f"  Error: Eye-tracking file is missing timestamp columns:
{TIMESTAMP_COLUMNS}. Skipping.")
            return None
    except Exception as e:
        print(f"  Error reading eye-tracking file {raw_tobii_filepath.name}: {e}. Skipping.")
        return None

    # Perform in-memory processing from Script 1
    et_df = add_aoi_columns(et_df, AOI_CATEGORIES)

    # Calculate timestamps (from Script 2)
    print("  Step 2: Calculating precise timestamps (ts_utc)...")
    try:
        et_df['start_dt_utc'] = pd.to_datetime(
```

```
            et_df['Recording date UTC'] + ' ' + et_df['Recording start time UTC'],
            format='%d-%m-%Y %H:%M:%S.%f', utc=True, errors='coerce'
        )
        et_df['ts_utc'] = et_df['start_dt_utc'] + pd.to_timedelta(
            pd.to_numeric(et_df['Recording timestamp'], errors='coerce'), unit='us'
        )
        et_df.dropna(subset=['ts_utc'], inplace=True)
        et_df = et_df.sort_values('ts_utc').reset_index(drop=True)
    except Exception as e:
        print(f"  Error calculating timestamps for {participant_id}: {e}. Skipping.")
        return None

    # 4. FILTER, MAP & MERGE (from Script 2)
    print(f"  Step 3: Filtering eye-tracking data to range: {overall_start} to {overall_end}")
    et_df_filtered = et_df[
        (et_df['ts_utc'] >= overall_start) & (et_df['ts_utc'] < overall_end)
    ].copy()

    if et_df_filtered.empty:
        print(f"  Warning: No eye-tracking data found within the defined timeframes for
{participant_id}.")
        return None

    print("  Step 4: Mapping eye-tracking samples to timeframe numbers...")
    et_df_filtered['classification_timeframe_number'] = pd.NA
    et_df_filtered['robot_appearance_timeframe_number'] = pd.NA

    for _, event_row in gaze_df.iterrows():
        # Map classification timeframe
        ct_mask = (et_df_filtered['ts_utc'] >= event_row['classification_time_start']) &
(et_df_filtered['ts_utc'] < event_row['classification_time_end'])
        et_df_filtered.loc[ct_mask, 'classification_timeframe_number'] =
event_row['classification_timeframe_number_val']

        # Map robot appearance timeframe
        if pd.notna(event_row['robot_appearance_time_start']):
            rat_mask = (et_df_filtered['ts_utc'] >= event_row['robot_appearance_time_start']) &
(et_df_filtered['ts_utc'] < event_row['robot_appearance_time_end'])
            et_df_filtered.loc[rat_mask, 'robot_appearance_timeframe_number'] =
event_row['robot_appearance_timeframe_number_val']

    et_df_filtered['ParticipantID'] = participant_id_col

    # Merge additional data from gaze log
    gaze_to_merge = gaze_df[['classification_timeframe_number_val'] +
GAZE_COLS_TO_MERGE].rename(
```

```
        columns={'classification_timeframe_number_val': 'classification_timeframe_number'}
    )
    final_df = pd.merge(et_df_filtered, gaze_to_merge, on='classification_timeframe_number',
how='left')

    # 5. FINALIZE AND RETURN
    # Ensure all required columns exist and are in the correct order
    for col in FINAL_OUTPUT_COLUMNS:
        if col not in final_df.columns:
            final_df[col] = pd.NA

    print(f"  Successfully processed participant {participant_id}. Found {len(final_df)} data
rows.")
    return final_df[FINAL_OUTPUT_COLUMNS]


# --- SCRIPT EXECUTION ---
if __name__ == "__main__":
    # Validate input directories
    if not EYETRACKING_INPUT_DIR.is_dir():
        print(f"Error: Eye-tracking input directory not found: {EYETRACKING_INPUT_DIR}")
        exit()
    if not GAZE_LOG_DIR.is_dir():
        print(f"Error: Gaze log directory not found: {GAZE_LOG_DIR}")
        exit()

    # Find gaze logs to drive the processing
    gaze_log_files = list(GAZE_LOG_DIR.glob("gaze_log_p*.csv"))
    if not gaze_log_files:
        print(f"No gaze log files found in {GAZE_LOG_DIR} matching 'gaze_log_p*.csv'.")
        exit()

    print(f"Found {len(gaze_log_files)} participant gaze logs to process.")

    all_participants_data = []

    # Main loop to process each participant
    for gaze_filepath in gaze_log_files:
        p_id = get_participant_id_from_filename(gaze_filepath.name)
        if not p_id:
            print(f"Could not extract participant ID from gaze file: {gaze_filepath.name}. Skipping.")
            continue

        # Find the matching raw eye-tracking file
        tobii_filepath = EYETRACKING_INPUT_DIR / f"eyetracking_{p_id}.tsv"
        if not tobii_filepath.exists():
```

```
            print(f"Warning: Matching eye-tracking file not found for {p_id} at {tobii_filepath}.
Skipping.")
            continue

        try:
            # Process this participant's data
            participant_df = process_single_participant(p_id, tobii_filepath, gaze_filepath)

            # If processing was successful, add the resulting DataFrame to our list
            if participant_df is not None and not participant_df.empty:
                all_participants_data.append(participant_df)
        except Exception as e:
            print(f"CRITICAL UNHANDLED ERROR processing participant {p_id}: {e}")
            import traceback
            traceback.print_exc()

    # Final combination step (from Script 3)
    if not all_participants_data:
        print("\n--- Processing Finished: No data was successfully processed for any participant. ---
")
    else:
        print(f"\n--- Combining data from {len(all_participants_data)} successfully processed
participants... ---")
        try:
            # Concatenate all the individual DataFrames into one master DataFrame
            master_df = pd.concat(all_participants_data, ignore_index=True)

            # Save the final combined data to a CSV file
            master_df.to_csv(FINAL_OUTPUT_CSV, sep=',', index=False, na_rep='NaN')

            print(f"\n✅ Success! Combined data saved to: '{FINAL_OUTPUT_CSV}'")
            print(f"   The final dataset has {master_df.shape[0]} rows and {master_df.shape[1]}
columns.")
            print("\n--- First 5 rows of the final combined data ---")
            print(master_df.head())
        except Exception as e:
            print(f"Error during final combination or saving: {e}")
```

R Script for Performance Analysis

```
# ---------------------------------------------------------------------------
# Script: statistics_performance_analysis.R
# Purpose: Load raw trial-level data (totalgaze.csv), process variables
#         for task performance (score, move duration), calculate extensive
#         descriptive statistics, conduct outlier checks for move duration.
#         Aggregate key performance DVs per participant per condition,
#         check ANOVA assumptions, perform 3x2 repeated measures ANOVA,
#         and visualize final results.
#
# UPDATED: This script now filters move_duration outliers based on a
#         2.5 SD rule per participant, analyzes accuracy as a percentage,
#         and generates a final bar chart for accuracy results.
#
# UPDATED AGAIN: Robot conditions renamed and reordered. Plots are now grouped
#          by difficulty within each robot condition.
# ---------------------------------------------------------------------------

# --- 1. SETUP: Load Necessary Packages ---
# install.packages(c("tidyverse", "patchwork", "scales", "rstatix", "ggpubr", "emmeans"))

library(tidyverse)
library(patchwork)
library(scales)
library(rstatix)
library(ggpubr)
library(emmeans)

# --- 2. LOAD DATA ---
file_path <- "totalgaze.csv"
data_raw <- NULL

cat(paste0("--- Attempting to load '", file_path, "' ---\n"))
tryCatch({
  data_raw <- read_csv(file_path)
  cat(paste0("--- Successfully loaded '", file_path, "'. ---\n"))
}, error = function(e) {
  cat(paste0("--- ERROR: Could not load '", file_path, "'. ---\n"))
  cat("Error message: ", e$message, "\n")
})
```

```
if (is.null(data_raw)) {
  stop("Script cannot proceed because data_raw was not loaded.")
}

cat("\n--- Initial Data Inspection (First few rows of raw data) ---\n"); print(head(data_raw))
cat("\n--- Initial Structure of the raw data (str) ---\n"); str(data_raw)

# --- 3. STANDARDIZE COLUMN NAMES & INITIAL TRANSFORMATIONS ---
participant_id_original_name <- "participant"
robot_col_original_name <- "Robot"
difficulty_input_col_name <- "difficulty"
correct_side_original_name <- "correct_side"
participants_side_choice_original_name <- "participants_side_choice"
move_duration_original_name <- "move_duration"

data <- data_raw

participant_id_col <- participant_id_original_name
robot_col <- robot_col_original_name
difficulty_original_col <- difficulty_input_col_name
correct_side_col <- correct_side_original_name
participant_choice_col <- participants_side_choice_original_name
move_duration_col <- move_duration_original_name

score_col <- "task_score"
difficulty_labelled_col <- "Difficulty_Condition"

if (difficulty_original_col %in% colnames(data)) {
  data <- data %>%
    mutate(
      !!sym(difficulty_original_col) := case_when(
        tolower(.data[[difficulty_original_col]]) == "easy" ~ 0,
        tolower(.data[[difficulty_original_col]]) == "hard" ~ 1,
        TRUE ~ NA_real_
      )
    )
  data[[difficulty_original_col]] <- as.numeric(data[[difficulty_original_col]])
}

if (correct_side_col %in% colnames(data) && participant_choice_col %in% colnames(data)) {
  data <- data %>%
    mutate(
      !!sym(score_col) := ifelse(
        is.na(!!sym(correct_side_col)) | is.na(!!sym(participant_choice_col)),
        NA_integer_,
```

```
      ifelse(as.character(!!sym(correct_side_col)) == as.character(!!sym(participant_choice_col)),
1, 0)
     )
   )
  data[[score_col]] <- as.integer(data[[score_col]])
}

# --- 4. PREPARE FACTORS & VERIFY COLUMN TYPES ---
cat("\n\n--- 4. Preparing IVs as Factors & Ensuring DV Numeric Types for Performance
Analysis ---\n")

if (robot_col %in% colnames(data) && !is.factor(data[[robot_col]])) {

  # <<< CHANGED: Renaming and reordering the robot conditions >>>
  # 1. First, rename the existing values to the new desired names.
  data <- data %>%
    mutate(!!sym(robot_col) := recode(!!sym(robot_col),
                                  "Ryan condition" = "Ryan (Joint)",
                                  "Ivan condition" = "Ivan (Disjoint)",
                                  "Carl condition" = "Carl (Control)"))

  # 2. Then, create the factor with the new names in the desired order.
  data[[robot_col]] <- factor(data[[robot_col]], levels = c("Ryan (Joint)", "Ivan (Disjoint)", "Carl
(Control)"))

  cat("--- Robot conditions have been renamed and reordered. New order: Ryan (Joint), Ivan
(Disjoint), Carl (Control) ---\n")
}

if (participant_id_col %in% colnames(data) && !is.factor(data[[participant_id_col]])) {
  data[[participant_id_col]] <- as.factor(data[[participant_id_col]])
}

if (difficulty_original_col %in% colnames(data) && is.numeric(data[[difficulty_original_col]]))
{
  data[[difficulty_labelled_col]] <- factor(data[[difficulty_original_col]], levels = c(0, 1), labels =
c("Easy", "Hard"))
} else if (difficulty_input_col_name %in% colnames(data) &&
is.character(data[[difficulty_input_col_name]])) {
  data[[difficulty_labelled_col]] <- factor(tolower(data[[difficulty_input_col_name]]), levels =
c("easy", "hard"), labels = c("Easy", "Hard"))
} else {
  stop(paste0("No usable difficulty column found to create the factor
'",difficulty_labelled_col,"'."))
}
```

```
performance_dvs_to_ensure_numeric <- c(score_col, move_duration_col)
for (dv_check in performance_dvs_to_ensure_numeric) {
  if (dv_check %in% colnames(data)) {
    if (!is.numeric(data[[dv_check]])) {
      data[[dv_check]] <- suppressWarnings(as.numeric(as.character(data[[dv_check]])))
    }
  }
}
cat("--- Performance data preparation complete. ---\n")

# --- 5. TRIAL-LEVEL OUTLIER HANDLING (for 'move_duration') ---
cat("\n\n--- 5. Trial-Level Outlier Visualization and Filtering for '", move_duration_col, "' ---")
if (move_duration_col %in% colnames(data) && is.numeric(data[[move_duration_col]])) {

  # --- 5.1 VISUALIZATION (using a Bar Chart of Means for Move Duration BEFORE
Filtering) ---
  cat(paste0("\n--- Visualizing Mean '", move_duration_col, "' with SD Error Bars (Trial-Level,
BEFORE Filtering) ---\n"))
  if (robot_col %in% colnames(data) && difficulty_labelled_col %in% colnames(data) ) {

    # Calculate summary stats for plotting move_duration
    summary_for_duration_plot <- data %>%
      filter(!is.na(!!sym(move_duration_col))) %>%
      group_by(!!sym(robot_col), !!sym(difficulty_labelled_col)) %>%
      summarise(
        Mean_Duration = mean(!!sym(move_duration_col), na.rm = TRUE),
        SD_Duration = sd(!!sym(move_duration_col), na.rm = TRUE),
        .groups = 'drop'
      )

    # <<< CHANGED: Plot structure updated to group by difficulty >>>
    md_by_condition_plot <- ggplot(summary_for_duration_plot,
                                   aes(x = !!sym(robot_col), y = Mean_Duration, fill =
!!sym(difficulty_labelled_col))) +
      geom_bar(stat = "identity", position = position_dodge(width = 0.9)) +
      geom_errorbar(aes(ymin = Mean_Duration - SD_Duration, ymax = Mean_Duration +
SD_Duration),
                    width = 0.25, position = position_dodge(width = 0.9)) +
      scale_fill_brewer(palette = "Pastel1") +
      labs(title = paste("Mean", move_duration_col, "by Condition (Before Outlier Filtering)"),
           subtitle = "Error bars represent +/- 1 Standard Deviation",
           y = paste("Mean", move_duration_col, "(seconds)"),
           x = "Robot Condition",
           fill = "Difficulty") +
      theme_minimal() +
```

```
    theme(legend.position = "top", axis.text.x = element_text(angle = 45, hjust = 1))
  print(md_by_condition_plot)
 }

 # --- 5.2 FILTERING (2.5 SD Rule per Participant for move_duration) ---
 cat(paste0("\n--- Filtering '", move_duration_col, "' outliers based on 2.5 SD rule per participant
---\n"))
 initial_rows <- nrow(data)
 cat(paste0("Initial number of trials: ", initial_rows, "\n"))

 data <- data %>%
   group_by(!!sym(participant_id_col)) %>%
   mutate(
     mean_dur = mean(!!sym(move_duration_col), na.rm = TRUE),
     sd_dur = sd(!!sym(move_duration_col), na.rm = TRUE),
     upper_bound = mean_dur + (2.5 * sd_dur),
     lower_bound = mean_dur - (2.5 * sd_dur)
   ) %>%
   filter(
     is.na(!!sym(move_duration_col)) | (!!sym(move_duration_col) >= lower_bound &
!!sym(move_duration_col) <= upper_bound)
   ) %>%
   ungroup() %>%
   select(-mean_dur, -sd_dur, -upper_bound, -lower_bound) # Clean up helper columns

 final_rows <- nrow(data)
 rows_removed <- initial_rows - final_rows
 percent_removed <- (rows_removed / initial_rows) * 100

 cat(paste0("Filtered number of trials: ", final_rows, "\n"))
 cat(paste0("Removed ", rows_removed, " trials (", round(percent_removed, 2), "%) as outliers
from '", move_duration_col, "'.\n"))

 # --- 5.3 VISUALIZATION (using a Bar Chart of Means for Move Duration AFTER Filtering)
---
 cat(paste0("\n--- Visualizing Mean '", move_duration_col, "' with SE Error Bars (Trial-Level,
AFTER Filtering) ---\n"))
 if (robot_col %in% colnames(data) && difficulty_labelled_col %in% colnames(data) ) {

   # Calculate summary stats for plotting move_duration from the CLEANED data
   summary_for_duration_plot_after <- data %>%
     filter(!is.na(!!sym(move_duration_col))) %>%
     group_by(!!sym(robot_col), !!sym(difficulty_labelled_col)) %>%
     summarise(
       Mean_Duration = mean(!!sym(move_duration_col), na.rm = TRUE),
       # Using Standard Error for the final plot is often better for inference
```

```
      SE_Duration = sd(!!sym(move_duration_col), na.rm = TRUE) / sqrt(n()),
      .groups = 'drop'
    )

  # <<< CHANGED: Plot structure updated to group by difficulty >>>
  md_by_condition_plot_after <- ggplot(summary_for_duration_plot_after,
                                      aes(x = !!sym(robot_col), y = Mean_Duration, fill =
!!sym(difficulty_labelled_col))) +
    geom_bar(stat = "identity", position = position_dodge(width = 0.9)) +
    # Error bars now represent +/- 1 Standard Error
    geom_errorbar(aes(ymin = Mean_Duration - SE_Duration, ymax = Mean_Duration +
SE_Duration),
                  width = 0.25, position = position_dodge(width = 0.9)) +
    scale_fill_brewer(palette = "Pastel1") +
    labs(title = paste("Mean", move_duration_col, "by Condition (After Outlier Filtering)"),
         subtitle = "Error bars represent +/- 1 Standard Error",
         y = paste("Mean", move_duration_col, "(seconds)"),
         x = "Robot Condition",
         fill = "Difficulty") +
    theme_minimal() +
    theme(legend.position = "top", axis.text.x = element_text(angle = 45, hjust = 1))

  print(md_by_condition_plot_after)
 }

} else {cat(paste0("\nNote: '", move_duration_col, "' column not found/specified or not numeric.
Outlier handling for move_duration skipped.\n"))}


# --- 6. DESCRIPTIVE STATISTICS (Trial-Level DVs on FILTERED data) ---
cat("\n\n--- 6. Descriptive Statistics (Trial-Level Performance DVs on Filtered Data) ---\n")
# 6.1 For 'task_score'
if (score_col %in% colnames(data) && is.numeric(data[[score_col]])) {
 cat(paste0("\n--- 6.1.1 Descriptive Statistics for '", score_col, "' by Robot x Difficulty ---\n"))
 descriptive_stats_score_crossed <- data %>% group_by(!!sym(robot_col),
!!sym(difficulty_labelled_col)) %>%
   summarise(N_trials = n(), Mean_Score_Prop = mean(!!sym(score_col), na.rm = TRUE),
SD_Score = sd(!!sym(score_col), na.rm = TRUE), .groups = 'drop')
 print(descriptive_stats_score_crossed)
}
# 6.2 For 'move_duration'
if (move_duration_col %in% colnames(data) && is.numeric(data[[move_duration_col]])) {
 cat(paste0("\n--- 6.2.1 Descriptive Statistics for '", move_duration_col, "' by Robot x Difficulty
---\n"))
 descriptive_stats_duration_crossed <- data %>% group_by(!!sym(robot_col),
!!sym(difficulty_labelled_col)) %>%
```

```
    summarise(N_trials = n(), Mean_Duration = mean(!!sym(move_duration_col), na.rm =
TRUE), SD_Duration = sd(!!sym(move_duration_col), na.rm = TRUE), .groups = 'drop')
  print(descriptive_stats_duration_crossed)
}

# --- 7. AGGREGATE PERFORMANCE DATA FOR ANOVA ---
cat("\n\n--- 7. Aggregating Performance Data per Participant for ANOVA ---\n")

data_agg_performance <- NULL # Initialize

if (nrow(data) > 0) {
  data_agg_performance <- data %>%
    group_by(!!sym(participant_id_col), !!sym(robot_col), !!sym(difficulty_labelled_col)) %>%
    summarise(
      Mean_Accuracy_Percent = if(score_col %in% colnames(.)) mean(!!sym(score_col), na.rm =
TRUE) * 100 else NA_real_,
      Mean_move_duration = if(move_duration_col %in% colnames(.))
mean(!!sym(move_duration_col), na.rm = TRUE) else NA_real_,
      N_Trials_Per_Condition = n(),
      .groups = 'drop'
    )

  cat("\n--- Aggregated Performance DVs for ANOVA (First few rows): ---\n")
  print(head(data_agg_performance))
  cat("\nStructure of aggregated Performance DVs for ANOVA:\n")
  str(data_agg_performance)
} else {
  stop("Error: No data remains after filtering. ANOVA cannot proceed.")
}


# --- 8. ANOVA DATA PREPARATION ---
cat("\n\n--- 8. Preparing Aggregated Data for ANOVA ---\n")

dv_accuracy_anova <- "Mean_Accuracy_Percent"
dv_duration_anova <- "Mean_move_duration"

if (!participant_id_col %in% colnames(data_agg_performance)) stop("Participant ID column
missing in aggregated data.")
if (!robot_col %in% colnames(data_agg_performance)) stop("Robot column missing in
aggregated data.")
if (!difficulty_labelled_col %in% colnames(data_agg_performance)) stop("Difficulty column
missing in aggregated data.")

if (!dv_accuracy_anova %in% colnames(data_agg_performance)) {
```

```
  warning(paste0("ANOVA DV '", dv_accuracy_anova, "' not found. Accuracy analyses
skipped."))
  dv_accuracy_anova <- NULL
}
if (!dv_duration_anova %in% colnames(data_agg_performance)) {
  warning(paste0("ANOVA DV '", dv_duration_anova, "' not found. Duration analyses
skipped."))
  dv_duration_anova <- NULL
}

# --- 9. ASSUMPTION CHECKING (Normality per cell) ---
check_normality_per_cell_anova <- function(df, dv_name, group1_name, group2_name) {
  if (is.null(dv_name) || !dv_name %in% colnames(df)) {
    cat(paste0("\nSkipping normality check: DV '", dv_name, "' not available.\n"))
    return()
  }
  cat(paste0("\n--- Normality Check for ANOVA DV: ", dv_name, " (within each ",
group1_name, " x ", group2_name, " cell) ---\n"))

  # <<< CHANGED: Updated facetting to match new plot style (group by robot) >>>
  hist_plot <- ggplot(df, aes(x = .data[[dv_name]])) +
    geom_histogram(aes(y = after_stat(density)), bins=10, fill = "skyblue", color = "black", alpha
= 0.7, na.rm = TRUE) +
    geom_density(alpha = .2, fill = "#FF6666", na.rm = TRUE) +
    facet_grid(as.formula(paste0("`", group2_name, "` ~ `", group1_name, "`")), scales = "free_y")
+
    labs(title = paste("Histograms of", dv_name, "(Aggregated)"), x = dv_name, y = "Density") +
theme_minimal()
  print(hist_plot)

  # <<< CHANGED: Updated facetting to match new plot style (group by robot) >>>
  qq_plot <- ggpubr::ggqqplot(df, x = dv_name, conf.int = TRUE, ggtheme = theme_minimal(),
title = paste("Q-Q Plots of", dv_name, "(Aggregated)")) +
    facet_grid(as.formula(paste0("`", group2_name, "` ~ `", group1_name, "`")), scales = "free")
  print(qq_plot)

  normality_tests <- df %>%
    group_by(!!sym(group1_name), !!sym(group2_name)) %>%
    filter(sum(!is.na(.data[[dv_name]])) >= 3) %>%
    summarise( shapiro_w = ifelse(sum(!is.na(.data[[dv_name]])) >=3,
shapiro.test(.data[[dv_name]])$statistic, NA_real_),
           shapiro_p = ifelse(sum(!is.na(.data[[dv_name]])) >=3,
shapiro.test(.data[[dv_name]])$p.value, NA_real_),
           n_for_test = sum(!is.na(.data[[dv_name]])), .groups = 'drop')
  cat("\n  Shapiro-Wilk Test Results (p > 0.05 suggests normality):\n"); print(normality_tests)
}
```

```
if (!is.null(dv_accuracy_anova)) { check_normality_per_cell_anova(data_agg_performance,
dv_accuracy_anova, robot_col, difficulty_labelled_col) }
if (!is.null(dv_duration_anova)) { check_normality_per_cell_anova(data_agg_performance,
dv_duration_anova, robot_col, difficulty_labelled_col) }

# --- 10. SIGNIFICANCE TESTING: 3x2 REPEATED MEASURES ANOVA ---
perform_rm_anova_integrated <- function(df, dv_col, wid_col, within_factors_cols) {
  if (is.null(dv_col) || !dv_col %in% colnames(df)) {
    cat(paste0("\nSkipping ANOVA: DV '", dv_col, "' not available.\n"))
    return(NULL)
  }
  cat(paste0("\n\n--- Repeated Measures ANOVA for: ", dv_col, " ---\n"))

  if(!is.numeric(df[[dv_col]])) {
    cat(paste0("  Warning: DV '", dv_col, "' is not numeric. Attempting conversion.\n"))
    df[[dv_col]] <- suppressWarnings(as.numeric(as.character(df[[dv_col]])))
    if(all(is.na(df[[dv_col]]))) {
      cat(paste0("  ERROR: DV '", dv_col, "' could not be converted to numeric or is all NA.
Skipping ANOVA.\n"))
      return(NULL)
    }
  }

  n_within_levels <- df %>% select(all_of(within_factors_cols)) %>% n_distinct()

  complete_cases_df <- df %>%
    filter(!is.na(.data[[dv_col]])) %>%
    group_by(!!sym(wid_col)) %>%
    filter(n() == n_within_levels) %>%
    ungroup()

  n_complete_subjects <- length(unique(complete_cases_df[[wid_col]]))

  if(n_complete_subjects < 2) {
    cat(paste0("  Warning: Not enough subjects (found ", n_complete_subjects, ") with complete
data for '", dv_col, "' across all conditions. Skipping ANOVA.\n"))
    return(NULL)
  }

  cat(paste0("  Performing ANOVA on ", n_complete_subjects, " participants with complete data
for ", dv_col, ".\n"))

  res_aov_obj <- NULL
  tryCatch({
    res_aov_obj <- anova_test(
```

```
      data = complete_cases_df,
      dv = !!sym(dv_col),
      wid = !!sym(wid_col),
      within = within_factors_cols
    )
    cat(paste0("\n  --- ANOVA Results for ", dv_col, " ---\n"))
    print(res_aov_obj)

    anova_table <- NULL
    if (is.list(res_aov_obj) && "ANOVA" %in% names(res_aov_obj)) {
      anova_table <- res_aov_obj$ANOVA
    } else if (is.data.frame(res_aov_obj) || is_tibble(res_aov_obj)) {
      anova_table <- res_aov_obj
    } else {
      cat("  Warning: Could not identify the ANOVA table within the anova_test result object.\n")
      return(res_aov_obj)
    }

    cat("\n  Key P-values and GES from ANOVA table:\n"); print(anova_table %>% filter(Effect
!= "(Intercept)") %>% select(Effect, p, ges))

    interaction_term_pattern <- paste(within_factors_cols, collapse=":")
    interaction_effect_row <- anova_table %>% filter(Effect == interaction_term_pattern)

    if (nrow(interaction_effect_row) == 1 && interaction_effect_row$p < 0.05) {
      cat(paste0("\n  --- Interaction effect '", interaction_term_pattern, "' for '", dv_col, "' was
significant (p = ", format(interaction_effect_row$p, digits=3),"). Probing simple effects... ---\n"))

      cat(paste0("  Simple main effect of ", within_factors_cols[1], " at each level of ",
within_factors_cols[2], ":\n"))
      simple_effects_1 <- complete_cases_df %>%
        group_by(!!sym(within_factors_cols[2])) %>%
        anova_test(formula = as.formula(paste0("`", dv_col, "` ~ `", within_factors_cols[1], "`")),
                   wid = !!sym(wid_col), within = !!sym(within_factors_cols[1])) %>%
        get_anova_table() %>%
        adjust_pvalue(method = "bonferroni")
      print(simple_effects_1)

      cat(paste0("\n  Simple main effect of ", within_factors_cols[2], " at each level of ",
within_factors_cols[1], ":\n"))
      simple_effects_2 <- complete_cases_df %>%
        group_by(!!sym(within_factors_cols[1])) %>%
        anova_test(formula = as.formula(paste0("`", dv_col, "` ~ `", within_factors_cols[2], "`")),
                   wid = !!sym(wid_col), within = !!sym(within_factors_cols[2])) %>%
        get_anova_table() %>%
        adjust_pvalue(method = "bonferroni")
```

```
    print(simple_effects_2)

    cat("\n  Consider further pairwise comparisons for significant simple effects with >2 levels using emmeans or pairwise_t_test.\n")

  } else {
    cat(paste0("\n  --- Interaction effect '", interaction_term_pattern, "' for '", dv_col, "' was NOT significant or not found. Checking main effects... ---\n"))

    main_effect_1_row <- anova_table %>% filter(Effect == within_factors_cols[1]) # Robot
    if (nrow(main_effect_1_row) == 1 && main_effect_1_row$p < 0.05) {
      cat(paste0("\n  --- Main effect of '", within_factors_cols[1], "' for '", dv_col, "' was significant (p = ", format(main_effect_1_row$p, digits=3), "). Pairwise comparisons (Bonferroni)... ---\n"))
      pwc_1 <- complete_cases_df %>%
        pairwise_t_test(as.formula(paste0("`", dv_col, "` ~ `", within_factors_cols[1], "`")),
                        paired = TRUE, p.adjust.method = "bonferroni")
      print(pwc_1)
    } else if (nrow(main_effect_1_row) == 1) {
      cat(paste0("\n  --- Main effect of '", within_factors_cols[1], "' for '", dv_col, "' was NOT significant (p = ", format(main_effect_1_row$p, digits=3), "). ---\n"))
    }

    main_effect_2_row <- anova_table %>% filter(Effect == within_factors_cols[2]) # Difficulty
    if (nrow(main_effect_2_row) == 1 && main_effect_2_row$p < 0.05) {
      cat(paste0("\n  --- Main effect of '", within_factors_cols[2], "' for '", dv_col, "' was significant (p = ", format(main_effect_2_row$p, digits=3), "). ---\n"))
      pwc_2 <- complete_cases_df %>%
        pairwise_t_test(as.formula(paste0("`", dv_col, "` ~ `", within_factors_cols[2], "`")),
                        paired = TRUE, p.adjust.method = "bonferroni")
      print(pwc_2)
    } else if (nrow(main_effect_2_row) == 1) {
      cat(paste0("\n  --- Main effect of '", within_factors_cols[2], "' for '", dv_col, "' was NOT significant (p = ", format(main_effect_2_row$p, digits=3), "). ---\n"))
    }
  }
  return(res_aov_obj)

}, error = function(e) {
  cat(paste0("  --- ERROR during Repeated Measures ANOVA for '", dv_col, "': ", e$message, " ---\n"))
  return(NULL)
})
}

# Perform ANOVA for Mean Accuracy
```

```
if (!is.null(data_agg_performance) && !is.null(dv_accuracy_anova)) {
  results_accuracy_anova <- perform_rm_anova_integrated(data_agg_performance,
dv_accuracy_anova, participant_id_col, c(robot_col, difficulty_labelled_col))
}

# Perform ANOVA for Mean Duration
if (!is.null(data_agg_performance) && !is.null(dv_duration_anova)) {
  results_duration_anova <- perform_rm_anova_integrated(data_agg_performance,
dv_duration_anova, participant_id_col, c(robot_col, difficulty_labelled_col))
}

cat("\n\n--- Performance Analysis Script (with ANOVA) Finished ---\n")

# --- 11. VISUALIZE AGGREGATED ACCURACY RESULTS ---
cat("\n\n--- 11. Visualizing Aggregated Accuracy Performance ---\n")
if (!is.null(data_agg_performance) && dv_accuracy_anova %in%
colnames(data_agg_performance)) {

  # Calculate summary statistics for the accuracy plot (Mean and Standard Error)
  accuracy_summary_for_plot <- data_agg_performance %>%
    group_by(!!sym(robot_col), !!sym(difficulty_labelled_col)) %>%
    summarise(
      Mean_Accuracy = mean(!!sym(dv_accuracy_anova), na.rm = TRUE),
      SE_Accuracy = sd(!!sym(dv_accuracy_anova), na.rm = TRUE) / sqrt(n()),
      .groups = 'drop'
    )
  cat("\nSummary statistics for accuracy plot:\n")
  print(accuracy_summary_for_plot)

  # <<< NOTE: This plot already had the correct structure and will update automatically with the
new names/order >>>
  # Create the bar chart for Mean Accuracy Percentage
  accuracy_plot <- ggplot(accuracy_summary_for_plot,
                        aes(x = !!sym(robot_col), y = Mean_Accuracy, fill =
!!sym(difficulty_labelled_col))) +
    geom_bar(stat = "identity", position = position_dodge(width = 0.9)) +
    geom_errorbar(aes(ymin = Mean_Accuracy - SE_Accuracy, ymax = Mean_Accuracy +
SE_Accuracy),
                  width = 0.25, position = position_dodge(width = 0.9)) +
    scale_fill_brewer(palette = "Pastel1") + # Using a color-blind friendly palette
    labs(title = "Mean Task Accuracy by Robot and Difficulty",
         x = "Robot Condition",
         y = "Mean Accuracy (%)",
         fill = "Difficulty") +
    theme_minimal(base_size = 14) +
    theme(legend.position = "top",
```

```
        axis.text.x = element_text(angle = 45, hjust = 1),
        plot.title = element_text(hjust = 0.5),
        panel.grid.major.x = element_blank(), # Cleaner look
        panel.grid.minor.y = element_blank()) +
    coord_cartesian(ylim = c(0, 100)) # Ensure Y axis goes from 0 to 100

  print(accuracy_plot)
  cat("\n--- Accuracy bar chart generated. ---\n")

} else {
  cat("\n--- Skipping accuracy bar chart: Aggregated data or accuracy DV not available. ---\n")
}

cat("\n--- Full Analysis Script Finished ---\n")
cat("Review all ANOVA tables, Mauchly's test results, post-hoc tests, and generated plots
carefully.\n")
```

Script for Gaze Follow Analysis

```
# ----------------------------------------------------------------------------
# Script: statistics_gaze_follow_analysis.R
# Purpose: Load raw trial-level data (totalgaze.csv), process variables
#          relevant to gaze following behavior, calculate descriptive
#          statistics with plots, and conduct inferential statistics
#          (GLMM and SDT) for gaze following.
# ----------------------------------------------------------------------------

# --- 1. SETUP: Load Necessary Packages ---
library(tidyverse)
library(lme4)      # For GLMM
library(car)       # For Anova function
library(emmeans)   # For post-hoc tests and plotting interactions
library(scales)    # For percent_format
library(afex)      # For repeated-measures ANOVA (for SDT)
library(patchwork) # For combining plots into a single figure

# --- 2. LOAD DATA ---
file_path <- "totalgaze.csv"
data_raw <- NULL

cat(paste0("--- Attempting to load '", file_path, "' ---\n"))
tryCatch({
  data_raw <- read_csv(file_path)
  cat(paste0("--- Successfully loaded '", file_path, "'. ---\n"))
}, error = function(e) {
  cat(paste0("--- ERROR: Could not load '", file_path, "'. ---\n"))
  cat("Error message: ", e$message, "\n")
})

if (is.null(data_raw)) {
  stop("Script cannot proceed because data_raw was not loaded.")
}

cat("\n--- Initial Data Inspection (First few rows of raw data) ---\n"); print(head(data_raw))

# --- 3. STANDARDIZE COLUMN NAMES & INITIAL TRANSFORMATIONS ---
participant_id_original_name <- "participant"
robot_col_original_name <- "Robot"
difficulty_input_col_name <- "difficulty"
correct_side_original_name <- "correct_side"
participants_side_choice_original_name <- "participants_side_choice"
gaze_decision_original_name <- "gazeDecision"
```

```
data_gaze_following <- data_raw

participant_id_col <- participant_id_original_name
robot_col <- robot_col_original_name
difficulty_original_col <- difficulty_input_col_name
correct_side_col <- correct_side_original_name
participant_choice_col <- participants_side_choice_original_name
gaze_decision_col <- gaze_decision_original_name

difficulty_labelled_col <- "Difficulty_Condition"

# --- 4. PREPARE FACTORS ---
cat("\n\n--- 4. Preparing IVs as Factors for Gaze Following Analysis ---\n")

if (robot_col %in% colnames(data_gaze_following) &&
!is.factor(data_gaze_following[[robot_col]])) {
  data_gaze_following[[robot_col]] <- factor(data_gaze_following[[robot_col]], levels = c("Carl
condition", "Ivan condition", "Ryan condition"))
  cat(paste0("Converted '", robot_col, "' to factor.\n"))
}
if (participant_id_col %in% colnames(data_gaze_following) &&
!is.factor(data_gaze_following[[participant_id_col]])) {
  data_gaze_following[[participant_id_col]] <-
as.factor(data_gaze_following[[participant_id_col]])
  cat(paste0("Converted '", participant_id_col, "' to factor.\n"))
}

if (difficulty_original_col %in% colnames(data_gaze_following)) {
  data_gaze_following[[difficulty_labelled_col]] <-
factor(tolower(data_gaze_following[[difficulty_original_col]]),
                                        levels = c("easy", "hard"),
                                        labels = c("Easy", "Hard"))
  cat(paste0("Created '", difficulty_labelled_col, "'.\n"))
} else {
  stop(paste0("Original difficulty column '",difficulty_original_col,"' not found."))
}
cat("--- Gaze following data preparation complete. ---\n")

# --- DATASET FOR MODELS (used in all subsequent analyses) ---
model_data_gaze <- data_gaze_following %>%
  filter(tolower(!!sym(gaze_decision_col)) %in% c("left", "right")) %>%
  mutate(
    robot_gaze_correct_val = ifelse(is.na(!!sym(gaze_decision_col)) |
is.na(!!sym(correct_side_col)), NA_integer_,
                          ifelse(as.character(!!sym(gaze_decision_col)) ==
as.character(!!sym(correct_side_col)), 1, 0)),
```

```
    robot_gaze_correct = factor(robot_gaze_correct_val, levels = c(0,1), labels = c("Incorrect
Gaze", "Correct Gaze")),
    gaze_followed_val = ifelse(is.na(!!sym(participant_choice_col)) |
is.na(!!sym(gaze_decision_col)), NA_integer_,
                            ifelse(as.character(!!sym(participant_choice_col)) ==
as.character(!!sym(gaze_decision_col)), 1, 0))
  ) %>%
  filter(!!sym(robot_col) %in% c("Ryan condition", "Ivan condition")) %>%
  filter(!is.na(gaze_followed_val) & !is.na(robot_gaze_correct) &
         !is.na(!!sym(difficulty_labelled_col)) & !is.na(!!sym(participant_id_col))) %>%
  mutate(
    Robot_Condition_Model = droplevels(factor(.data[[robot_col]])),
    Difficulty_Model = factor(.data[[difficulty_labelled_col]]),
    Participant_ID_Model = factor(.data[[participant_id_col]]),
    Gaze_Correctness_Model = factor(robot_gaze_correct)
  )

# --- 5. DESCRIPTIVE STATISTICS & VISUALIZATION ---
cat("\n\n--- 5. Generating Descriptive Statistics and Plots ---\n")

if (exists("model_data_gaze") && nrow(model_data_gaze) > 0) {
  # Calculate counts for each condition (Easy/Hard)
  descriptive_summary <- model_data_gaze %>%
    group_by(Robot_Condition_Model, Gaze_Correctness_Model, Difficulty_Model) %>%
    summarise(
      n_followed = sum(gaze_followed_val, na.rm = TRUE),
      n_total_trials = n(),
      .groups = 'drop'
    )

  # --- [NEW] 5.0.1 DISPLAY DESCRIPTIVE PERCENTAGES IN CONSOLE ---
  cat("\n\n--- 5.0.1. Gaze Following Percentages by Condition ---\n")

  # Calculate and format percentages for clear console output
  descriptive_percentages <- descriptive_summary %>%
    mutate(
      percentage_followed = (n_followed / n_total_trials),
      # Format for printing
      percentage_str = scales::percent(percentage_followed, accuracy = 0.1),
      # Relabel robot conditions to "Joint" and "Disjoint" for clarity
      Robot_Condition_Display = case_when(
        Robot_Condition_Model == "Ryan condition" ~ "Joint (Ryan)",
        Robot_Condition_Model == "Ivan condition" ~ "Disjoint (Ivan)",
        TRUE ~ as.character(Robot_Condition_Model)
      )
    ) %>%
```

```
  # Select and reorder columns for a clean table view
  select(
    Robot_Condition_Display,
    Difficulty_Model,
    Gaze_Correctness_Model,
    percentage_str,
    n_followed,
    n_total_trials
  ) %>%
  # Arrange for easy reading
  arrange(Robot_Condition_Display, Difficulty_Model, Gaze_Correctness_Model)

print(descriptive_percentages, n = Inf) # n = Inf ensures all rows are printed
# --- [END NEW SECTION] ---

# --- 5.1 Visualization of Descriptive Statistics ---
cat("\n\n--- 5.1. Creating Bar Charts for Descriptive Gaze Following ---\n")

# Prepare data for plotting by calculating percentages and relabeling
descriptive_plot_data <- descriptive_summary %>%
  mutate(
    percentage_followed = (n_followed / n_total_trials),
    # Relabel robot conditions to "Joint" and "Disjoint"
    Robot_Condition_Plot = case_when(
      Robot_Condition_Model == "Ryan condition" ~ "Joint",
      Robot_Condition_Model == "Ivan condition" ~ "Disjoint",
      TRUE ~ as.character(Robot_Condition_Model)
    )
  ) %>%
  # Set the order for the factor so the legend and colors are correct
  mutate(Robot_Condition_Plot = factor(Robot_Condition_Plot, levels = c("Joint", "Disjoint")))

# Custom theme to match the python plot style
theme_custom_style <- function() {
  theme_minimal(base_size = 12) +
    theme(
      plot.title = element_text(hjust = 0.5, face = "bold"), # Center title
      panel.border = element_rect(colour = "black", fill=NA, linewidth=1), # Add border
      panel.grid.major.x = element_blank(),
      panel.grid.minor.x = element_blank(),
      panel.grid.major.y = element_line(linetype = "dashed", color = "grey80"),
      panel.grid.minor.y = element_blank(),
      legend.title = element_blank(), # Remove legend title
      axis.title.x = element_blank() # Remove x-axis title from individual plots
    )
}
```

```
  # Plot 1: Gaze Following after CORRECT Gaze Cues
  plot_desc_correct <- descriptive_plot_data %>%
    filter(Gaze_Correctness_Model == "Correct Gaze") %>%
    ggplot(aes(x = Difficulty_Model, y = percentage_followed, fill = Robot_Condition_Plot)) +
    geom_bar(stat = "identity", position = position_dodge(0.8), width = 0.7) +
    scale_y_continuous(labels = scales::percent_format(accuracy=1), limits = c(0, 1.01), expand =
c(0, 0)) +
    scale_fill_manual(values = c("Joint" = "skyblue", "Disjoint" = "steelblue")) +
    labs(title = "Correct Gaze Following", y = "Percentage") +
    theme_custom_style()

  # Plot 2: Gaze Following after INCORRECT Gaze Cues
  plot_desc_incorrect <- descriptive_plot_data %>%
    filter(Gaze_Correctness_Model == "Incorrect Gaze") %>%
    ggplot(aes(x = Difficulty_Model, y = percentage_followed, fill = Robot_Condition_Plot)) +
    geom_bar(stat = "identity", position = position_dodge(0.8), width = 0.7) +
    scale_y_continuous(labels = scales::percent_format(accuracy=1), limits = c(0, 1.01), expand =
c(0, 0)) +
    scale_fill_manual(values = c("Joint" = "lightcoral", "Disjoint" = "indianred")) +
    labs(title = "Incorrect Gaze Following", y = NULL) + # Remove y-axis title for shared axis
    theme_custom_style()

  # Combine the plots side-by-side using patchwork
  combined_plot <- plot_desc_correct + plot_desc_incorrect +
    plot_annotation(
      title = 'Gaze Following Behavior (Easy vs Hard)',
      theme = theme(plot.title = element_text(hjust = 0.5, size = 16, face = "bold"))
    )

  cat("\n--- Displaying Combined Descriptive Plot ---\n")
  print(combined_plot)


} else {
  cat("\n--- Skipping Descriptive Statistics & Plots: 'model_data_gaze' not available or empty. ---
\n")
}

# --- 6. COMPLEMENTARY GLMM ANALYSIS (FOR APPENDIX) ---
cat("\n\n--- 6. Complementary GLMM Analysis (For Appendix) ---\n")

if (nrow(model_data_gaze) > 50 && n_distinct(model_data_gaze$Participant_ID_Model) > 1) {
  options(contrasts = c("contr.sum", "contr.poly"))
  gaze_follow_glmm <- NULL
  tryCatch({
```

```
    formula_str <- "gaze_followed_val ~ Robot_Condition_Model * Gaze_Correctness_Model *
Difficulty_Model + (1 | Participant_ID_Model)"
    gaze_follow_glmm <- glmer( as.formula(formula_str), data = model_data_gaze,
                            family = binomial(link = "logit"), control = glmerControl(optimizer =
"bobyqa", optCtrl = list(maxfun = 2e5)))
    cat("--- GLMM fitting successful. ---\n")
    cat("\n--- ANOVA Table (Type III Wald Chi-square tests) for GLMM ---\n")
    print(Anova(gaze_follow_glmm, type = "III"))
  }, error = function(e) { cat("--- ERROR during GLMM fitting: ---\n"); print(e) })
}

# --- 7. PRIMARY ANALYSIS: SIGNAL DETECTION THEORY (SDT) ---
cat("\n\n\n--- 7. Primary Analysis: Signal Detection Theory (SDT) ---\n")

if (exists("model_data_gaze") && nrow(model_data_gaze) > 0) {

  # --- 7.1. Calculate SDT Counts ---
  cat("\n--- 7.1. Calculating SDT counts per participant and condition ---\n")
  sdt_counts <- model_data_gaze %>%
    mutate(
      sdt_outcome = case_when(
        gaze_followed_val == 1 & robot_gaze_correct_val == 1 ~ "Hit",
        gaze_followed_val == 0 & robot_gaze_correct_val == 1 ~ "Miss",
        gaze_followed_val == 1 & robot_gaze_correct_val == 0 ~ "False Alarm",
        gaze_followed_val == 0 & robot_gaze_correct_val == 0 ~ "Correct Rejection"
      )
    ) %>%
    group_by(Participant_ID_Model, Robot_Condition_Model, Difficulty_Model) %>%
    summarise(
      n_hits = sum(sdt_outcome == "Hit", na.rm = TRUE),
      n_misses = sum(sdt_outcome == "Miss", na.rm = TRUE),
      n_fas = sum(sdt_outcome == "False Alarm", na.rm = TRUE),
      n_crs = sum(sdt_outcome == "Correct Rejection", na.rm = TRUE),
      .groups = 'drop'
    )

  # --- 7.2. Calculate d' and c ---
  cat("\n--- 7.2. Calculating d' (sensitivity) and c (criterion) ---\n")
  sdt_results <- sdt_counts %>%
    mutate(
      # Apply log-linear correction to prevent infinite values
      H = (n_hits + 0.5) / (n_hits + n_misses + 1),
      FA = (n_fas + 0.5) / (n_fas + n_crs + 1),
      d_prime = qnorm(H) - qnorm(FA),
      criterion_c = -0.5 * (qnorm(H) + qnorm(FA))
    )
```

```
# --- 7.3. Inferential Statistics on d' and c ---
cat("\n--- 7.3. Running Repeated Measures ANOVAs on d' and c ---\n")

# Analysis 1: Sensitivity (d').
cat("\n--- ANOVA on d' (Sensitivity) ---\n")
anova_d_prime <- aov_ez(
  id = "Participant_ID_Model", dv = "d_prime", data = sdt_results,
  within = c("Robot_Condition_Model", "Difficulty_Model")
)
print(summary(anova_d_prime))

# Analysis 2: Bias (c).
cat("\n--- ANOVA on c (Bias/Criterion) ---\n")
anova_criterion_c <- aov_ez(
  id = "Participant_ID_Model", dv = "criterion_c", data = sdt_results,
  within = c("Robot_Condition_Model", "Difficulty_Model")
)
print(summary(anova_criterion_c))

# --- 7.4. Post-Hoc Analysis for Significant Main Effects ---
cat("\n--- 7.4. Post-Hoc analysis for significant main effect of Robot on Criterion (c) ---\n")
emm_c_robot <- emmeans(anova_criterion_c, ~ Robot_Condition_Model)
print(summary(emm_c_robot))

# --- 7.5. Visualization of SDT Results ---
cat("\n--- 7.5. Creating Bar Charts for d' and c ---\n")

# Create a summary dataframe with means and CIs for plotting
sdt_summary_for_plotting <- sdt_results %>%
  group_by(Robot_Condition_Model, Difficulty_Model) %>%
  summarise(
    mean_d_prime = mean(d_prime, na.rm = TRUE),
    se_d_prime = sd(d_prime, na.rm = TRUE) / sqrt(n()),
    ci_d_prime = se_d_prime * qt(0.975, df = n() - 1),
    mean_c = mean(criterion_c, na.rm = TRUE),
    se_c = sd(criterion_c, na.rm = TRUE) / sqrt(n()),
    ci_c = se_c * qt(0.975, df = n() - 1),
    .groups = 'drop'
  ) %>%
  # RENAME AND REORDER the Robot Condition factor for plotting
  mutate(
    Robot_Condition_Model = case_when(
      Robot_Condition_Model == "Ryan condition" ~ "Joint Condition",
      Robot_Condition_Model == "Ivan condition" ~ "Disjoint Condition",
      TRUE ~ as.character(Robot_Condition_Model)
```

```
    ),
    Robot_Condition_Model = factor(Robot_Condition_Model, levels = c("Joint Condition",
"Disjoint Condition"))
  )

 # Plot 1: Sensitivity (d') - Now with updated names and order
 plot_d_prime <- ggplot(sdt_summary_for_plotting,
                        aes(x = Difficulty_Model, y = mean_d_prime, fill = Robot_Condition_Model)) +
   geom_bar(stat = "identity", position = position_dodge(0.9), color = "black", width = 0.8) +
   geom_errorbar(aes(ymin = mean_d_prime - ci_d_prime, ymax = mean_d_prime +
ci_d_prime),
                 position = position_dodge(0.9), width = 0.25, linewidth = 0.5) +
   scale_fill_brewer(palette = "Pastel1", name = "Robot Condition") +
   labs(title = "Sensitivity to Gaze Cue Validity",
        subtitle = "Participants' ability to discriminate correct from incorrect gaze cues.",
        x = "Task Difficulty",
        y = "Sensitivity (d')") +
   theme_minimal(base_size = 14) +
   theme(legend.position = "top",
         plot.title = element_text(face = "bold"),
         axis.title = element_text(face = "bold"))

 cat("\n--- Displaying Sensitivity (d') Plot ---\n")
 print(plot_d_prime)

 # Plot 2: Response Criterion (c) - Now with updated names and order
 plot_criterion_c <- ggplot(sdt_summary_for_plotting,
                            aes(x = Robot_Condition_Model, y = mean_c, fill = Difficulty_Model)) +
   geom_bar(stat = "identity", position = position_dodge(0.9), color = "black", width = 0.8) +
   geom_errorbar(aes(ymin = mean_c - ci_c, ymax = mean_c + ci_c),
                 position = position_dodge(0.9), width = 0.25, linewidth = 0.5) +
   geom_hline(yintercept = 0, linetype = "dashed", color = "grey30") +
   scale_fill_brewer(palette = "Pastel2", name = "Task Difficulty") +
   labs(title = "Response Bias for Following Gaze Cues",
        subtitle = "A negative value indicates a liberal bias (tendency to follow).",
        x = "Robot Condition",
        y = "Response Criterion (c)") +
   theme_minimal(base_size = 14) +
   theme(legend.position = "top",
         plot.title = element_text(face = "bold"),
         axis.title = element_text(face = "bold"))

 cat("\n--- Displaying Response Criterion (c) Plot ---\n")
 print(plot_criterion_c)

} else {
```

```
  cat("\n--- Skipping SDT analysis: 'model_data_gaze' not available or empty. ---\n")
}

cat("\n\n--- End of script processing. ---\n")
```

Script for Qualtrics Analysis

```
# ---------------------------------------------------------------------------
# FULLY CONSOLIDATED SCRIPT FOR QUALTRICS SUBJECTIVE DATA ANALYSIS
(v2)
# Combines:
# 1. Loading of qualtrics_data_final.csv and robust participantID handling
# 2. Demographic Variable Processing (Age, Gender, Residence)
# 3. Calculation of Composite Scale Scores (Anthro, Like, Intel, Trust for Carl, Ryan, Ivan)
# 4. Descriptive Statistics & ALL Visualizations (including new faceted Bar Chart)
# 5. Outlier Identification for Composite Scores
# 6. Normality Assumption Checks for Composite Scores
# 7. Reliability Analysis (Cronbach's Alpha)
# 8. Parametric Testing (Repeated Measures ANOVA for each construct)
# 9. Saving final dataset with all processed data and composite scores
# ---------------------------------------------------------------------------

# --- 1. SETUP: Load Necessary Packages ---
# Ensure these are installed by running install.packages("package_name") in your console once.
library(tidyverse)
library(lubridate) # For date parsing (Age)
library(psych)     # For Cronbach's Alpha
library(rstatix)   # For anova_test and other convenient stats functions
library(ggpubr)    # For ggqqplot

# --- 2. LOAD INITIAL RAW DATA ---
file_path <- "qualtrics_data_final.csv"
data <- NULL

cat(paste0("--- Attempting to load '", file_path, "' ---\n"))
tryCatch({
  data <- read_csv(file_path)
  cat(paste0("--- Successfully loaded '", file_path, "'. ---\n"))
}, error = function(e) {
  cat(paste0("--- ERROR: Could not load '", file_path, "'. ---\n"))
  cat("Error message: ", e$message, "\n")
})

if (is.null(data)) {
  stop("Script cannot proceed because data was not loaded.")
}

# --- Handle Participant ID ---
# Robustly find and set the participant ID column, which is essential for repeated measures
ANOVA.
if ("participant" %in% colnames(data) && !"participantID" %in% colnames(data)) {
```

```
  data <- data %>% rename(participantID = participant)
  cat("Renamed 'participant' column to 'participantID' for compatibility.\n")
} else if (!"participantID" %in% colnames(data)) {
  data$participantID <- 1:nrow(data)
  cat("Warning: No 'participant' or 'participantID' column found. Created a new 'participantID'
column.\n")
}


cat("\n--- Initial Data Inspection (First few rows) ---\n"); print(head(data))
cat("\n--- Initial Structure of the data (str) ---\n"); str(data)
cat("\n--- Initial Summary of the data ---\n"); print(summary(data))
cat("\n--- Column names in the loaded data: ---\n"); print(colnames(data))


# --- 3. PROCESS DEMOGRAPHIC VARIABLES ---
cat("\n\n--- 3. Processing Demographic Variables ---\n")

# --- 3.1 Variable: Age ---
cat("\n--- Processing 'Age' (Date of Birth) Column ---\n")
if ("Age" %in% colnames(data)) {
  data$DOB_original <- data$Age # Keep a copy

  cat("Attempting to parse DOBs with multiple formats...\n")
  data$DOB_parsed <- parse_date_time(data$DOB_original,
                                     orders = c(
                                       "d-m-Y", "d/m/Y", "d.m.Y",
                                       "d-m-y", "d/m/y", "d.m.y",
                                       "m-d-Y", "m/d/Y", "m-d-y", "m/d/y",
                                       "Y-m-d", "Y/m/d", "Ymd",
                                       "d-m-Y HMS", "d/m/Y HMS", "Y-m-d HMS", "Y/m/d HMS"
                                     ),
                                     quiet = TRUE)

  parsed_count <- sum(!is.na(data$DOB_parsed))
  total_count <- nrow(data)
  cat(paste0(parsed_count, " out of ", total_count, " DOBs successfully parsed.\n"))

  if (parsed_count < total_count) {
    failed_to_parse <- data$DOB_original[is.na(data$DOB_parsed) &
!is.na(data$DOB_original)]
    if(length(failed_to_parse) > 0) {
      cat("DOBs that failed to parse (first few shown):\n"); print(head(failed_to_parse))
    }
  }
```

```
  data$Age_years <- NA
  valid_dob_indices <- !is.na(data$DOB_parsed)

  if(any(valid_dob_indices)) {
    data$Age_years[valid_dob_indices] <- floor(as.numeric(difftime(Sys.Date(),
data$DOB_parsed[valid_dob_indices], units = "days")) / 365.25)

    cat("\n--- Descriptive Statistics for Calculated Age (in years) ---\n")
    print(summary(data$Age_years))
    cat("Standard Deviation of Age (years):", sd(data$Age_years, na.rm = TRUE), "\n")

    if (sum(!is.na(data$Age_years)) > 0) {
      cat("Generating Age Histogram...\n")
      print(ggplot(data[!is.na(data$Age_years), ], aes(x = Age_years)) +
              geom_histogram(binwidth = 1, fill = "skyblue", color = "black") +
              labs(title = "Distribution of Calculated Age", x = "Age (Years)", y = "Frequency") +
              theme_minimal())
      cat("Generating Age Boxplot...\n")
      print(ggplot(data[!is.na(data$Age_years), ], aes(y = Age_years)) +
              geom_boxplot(fill = "skyblue") +
              labs(title = "Boxplot of Calculated Age", y = "Age (Years)") +
              theme_minimal() + coord_flip())
    }
  } else {
    cat("No DOBs were successfully parsed, so 'Age_years' could not be calculated.\n")
  }
} else {
  cat("\n'Age' column (for DOB) not found in the dataset.\n")
}

# --- 3.2 Variable: Gender ---
cat("\n\n--- Processing 'gender' Column ---\n")
if ("gender" %in% colnames(data)) {
  if (!is.factor(data$gender)) { data$gender <- as.factor(data$gender) }
  cat("\n--- Frequency Table for Gender ---\n")
  gender_counts <- table(data$gender, useNA = "ifany")
  gender_percentages <- prop.table(gender_counts) * 100
  gender_levels <- names(gender_counts); if (any(is.na(gender_levels))) {
gender_levels[is.na(gender_levels)] <- "NA_Category" }
  if (length(gender_counts) > 0) {
    gender_summary_df <- data.frame(Category = gender_levels, Count =
as.integer(gender_counts), Percentage = as.numeric(gender_percentages))
    print(gender_summary_df)
  }
  if (sum(!is.na(data$gender)) > 0) {
    cat("Generating Gender Bar Chart...\n")
```

```
    print(ggplot(data[!is.na(data$gender), ], aes(x = gender, fill = gender)) +
            geom_bar(show.legend = FALSE) +
            geom_text(stat='count', aes(label=after_stat(count)), vjust=-0.5, size=3) +
            labs(title = "Distribution of Gender", x = "Gender", y = "Count") +
            theme_minimal() + theme(axis.text.x = element_text(angle = 45, hjust = 1)))
  }
} else {
  cat("\n'gender' column not found.\n")
}

# --- 3.3 Variable: Country of Residence ---
cat("\n\n--- Processing 'residence' Column ---\n")
if ("residence" %in% colnames(data)) {
  if (!is.factor(data$residence)) { data$residence <- as.factor(data$residence) }
  cat("\n--- Frequency Table for Residence ---\n")
  residence_counts <- table(data$residence, useNA = "ifany")
  residence_percentages <- prop.table(residence_counts) * 100
  category_names_res <- names(residence_counts)
  if (any(is.na(category_names_res))) { category_names_res[is.na(category_names_res)] <- "NA
(Missing)" } # Handles NA level name

  if (length(residence_counts) > 0) {
    residence_summary_df <- data.frame(
      Category = category_names_res,
      Count = as.integer(residence_counts),
      Percentage = as.numeric(residence_percentages)
    )
    residence_summary_df <- residence_summary_df[order(-residence_summary_df$Count), ]

    cat("\nSummary Table for Residence:\n")
    print(residence_summary_df, row.names = FALSE)
  } else {
    cat("No data (or only NA values) found in 'residence' column to create summary table.\n")
  }
} else {
  cat("\n'residence' column not found.\n")
}
cat("\n--- Demographic processing finished. ---\n")

# --- 4. DEFINE ITEMS FOR EACH SCALE AND ROBOT ---
cat("\n\n--- 4. Defining Scale Items ---\n")
anthro_carl_items <- paste0("anthropomorphism _", 1:5, "_carl"); like_carl_items <-
paste0("likability _", 1:5, "_carl"); intel_carl_items <- paste0("intelligence _", 1:5, "_carl");
trust_carl_items <- paste0("trust _", 1:14, "_carl")
```

```
anthro_ryan_items <- paste0("anthropomorphism _", 1:5, "_ryan"); like_ryan_items <-
paste0("likability _", 1:5, "_ryan"); intel_ryan_items <- paste0("intelligence _", 1:5, "_ryan");
trust_ryan_items <- paste0("trust _", 1:14, "_ryan")
anthro_ivan_items <- paste0("anthropomorphism _", 1:5, "_ivan"); like_ivan_items <-
paste0("likability _", 1:5, "_ivan"); intel_ivan_items <- paste0("intelligence _", 1:5, "_ivan");
trust_ivan_items <- paste0("trust _", 1:14, "_ivan")

# --- 5. CALCULATE COMPOSITE SCORES (ROW MEANS) ---
cat("\n\n--- 5. Calculating Composite Scores ---\n")
check_and_calculate_mean <- function(df, items_list, new_col_name) {
  existing_items <- intersect(items_list, colnames(df))
  if (length(existing_items) == 0) { cat(paste0("Warning: No items for '", new_col_name, "'.
Skipping.\n")); return(df) }
  if (length(existing_items) < length(items_list)) { cat(paste0("Warning: Not all items for '",
new_col_name, "' found. Using: ", paste(existing_items, collapse=", "), "\n")) }
  df <- df %>% mutate(across(all_of(existing_items), as.numeric)) # Ensure items are numeric
before rowMeans
  df <- df %>% mutate(!!new_col_name := rowMeans(select(., all_of(existing_items)), na.rm =
TRUE))
  cat(paste0("Calculated: ", new_col_name, "\n"))
  return(df)
}
data <- check_and_calculate_mean(data, anthro_carl_items, "Anthro_Carl_Score"); data <-
check_and_calculate_mean(data, like_carl_items, "Like_Carl_Score"); data <-
check_and_calculate_mean(data, intel_carl_items, "Intel_Carl_Score"); data <-
check_and_calculate_mean(data, trust_carl_items, "Trust_Carl_Score")
data <- check_and_calculate_mean(data, anthro_ryan_items, "Anthro_Ryan_Score"); data <-
check_and_calculate_mean(data, like_ryan_items, "Like_Ryan_Score"); data <-
check_and_calculate_mean(data, intel_ryan_items, "Intel_Ryan_Score"); data <-
check_and_calculate_mean(data, trust_ryan_items, "Trust_Ryan_Score")
data <- check_and_calculate_mean(data, anthro_ivan_items, "Anthro_Ivan_Score"); data <-
check_and_calculate_mean(data, like_ivan_items, "Like_Ivan_Score"); data <-
check_and_calculate_mean(data, intel_ivan_items, "Intel_Ivan_Score"); data <-
check_and_calculate_mean(data, trust_ivan_items, "Trust_Ivan_Score")
cat("\n--- Composite score calculation finished. ---\n")

# --- 6. DESCRIPTIVE STATISTICS & VISUALIZATIONS OF COMPOSITE SCORES ---
cat("\n\n--- 6. Descriptive Statistics & Visualizations of Composite Scores ---\n")
composite_score_columns <- c(grep("_Score$", colnames(data), value = TRUE)) # Dynamically
get all score columns
existing_composite_score_columns <- intersect(composite_score_columns, colnames(data))

if(length(existing_composite_score_columns) > 0){
  data <- data %>% mutate(across(all_of(existing_composite_score_columns), as.numeric)) #
Ensure numeric
  cat("\n--- Summary (Min, Q1, Median, Mean, Q3, Max) for Composite Scores ---\n")
```

```
  print(summary(data[, existing_composite_score_columns]))
  cat("\n--- Mean, SD, N for Composite Scores ---\n")
  desc_stats_mean_sd <- data %>%
    select(all_of(existing_composite_score_columns)) %>%
    pivot_longer(cols = everything(), names_to = "Score_Name", values_to = "Value") %>%
    group_by(Score_Name) %>%
    summarise(Mean = mean(Value, na.rm = TRUE), SD = sd(Value, na.rm = TRUE), N_obs =
sum(!is.na(Value))) %>%
    arrange(Score_Name)
  print(desc_stats_mean_sd, n = Inf)
} else { cat("No composite score columns found to summarize.\n")}

# --- Define names and orders for plotting ---
construct_name_map <- c(Anthro = "Anthropomorphism", Like = "Likability", Intel =
"Intelligence", Trust = "Trust")
constructs_short_names_for_iteration <- names(construct_name_map)

# This order is used for original plots and for the ANOVA calculations later
robots_order <- c("Carl", "Ryan", "Ivan")

# --- Define new orders, display names, and colors for the faceted plots ---
robots_order_original_faceted <- c("Ryan", "Ivan", "Carl")
robot_display_names_faceted <- c("Joint", "Disjoint", "Control")
robot_colors_faceted <- c("Joint" = "#029e73", "Disjoint" = "#d55e00", "Control" = "#cc78bc")


cat("\n\n--- 6.1 Generating Individual Boxplots for Each Composite Score ---\n")
if (length(existing_composite_score_columns) > 0) {
  for (score_col_indiv_plot in existing_composite_score_columns) {
    if (sum(!is.na(data[[score_col_indiv_plot]])) > 0) {
      short_construct_name_plot <- str_extract(score_col_indiv_plot, "^(Anthro|Like|Intel|Trust)")
      robot_name_plot <- str_extract(score_col_indiv_plot, "(Carl|Ryan|Ivan)")
      full_construct_display_name_plot <- construct_name_map[[short_construct_name_plot]]
      plot_title_desc <- paste("Boxplot of", full_construct_display_name_plot, "(Robot:",
robot_name_plot, ")")
      if (is.na(full_construct_display_name_plot) || is.na(robot_name_plot)) { plot_title_desc <-
paste("Boxplot of", score_col_indiv_plot) }

      p_i <- ggplot(data, aes(y = .data[[score_col_indiv_plot]])) +
        geom_boxplot(fill = "skyblue", outlier.colour = "red", outlier.shape = 16, outlier.size = 2) +
        labs(title = plot_title_desc, y = "Score", x = "") + theme_minimal() + theme(axis.text.x =
element_blank(), axis.ticks.x = element_blank())
      print(p_i); cat(paste("Boxplot for:", score_col_indiv_plot, "\n"))
    } else { cat(paste("Skipping boxplot for:", score_col_indiv_plot, "- All values are NA.\n"))}
  }
}
```

```
cat("\n\n--- 6.2 Generating Grouped Boxplots for Each Construct ---\n")
if ("participantID" %in% colnames(data)) {
  for (short_construct_name_grp_plot in constructs_short_names_for_iteration) {
    full_construct_display_name_grp_plot <-
construct_name_map[[short_construct_name_grp_plot]]
    cat(paste0("\n--- Grouped Boxplot for: ", full_construct_display_name_grp_plot, " Scores ---
\n"))
    score_cols_for_grp_plot <- existing_composite_score_columns[grep(paste0("^",
short_construct_name_grp_plot, "_"), existing_composite_score_columns)]
    if (length(score_cols_for_grp_plot) > 0) {
      data_long_construct_grp_plot <- data %>%
        select(participantID, all_of(score_cols_for_grp_plot)) %>%
        pivot_longer(cols = all_of(score_cols_for_grp_plot), names_to = "Scale_Version",
values_to = "Score") %>%
        mutate(Robot = str_extract(Scale_Version, paste(robots_order, collapse="|")), Robot =
factor(Robot, levels = robots_order))
      if (sum(!is.na(data_long_construct_grp_plot$Score)) > 0) {
        grouped_plot_render <- ggplot(data_long_construct_grp_plot, aes(x = Robot, y = Score, fill
= Robot)) +
          geom_boxplot(outlier.colour = "red", outlier.shape = 16, outlier.size = 2) +
          labs(title = paste(full_construct_display_name_grp_plot, "Scores by Robot"), x = "Robot",
y = paste(full_construct_display_name_grp_plot, "Score")) +
          theme_minimal() + theme(legend.position = "none")
        print(grouped_plot_render)
      } else { cat(paste("No non-NA data for grouped", full_construct_display_name_grp_plot,
"boxplot.\n")) }
    } else { cat(paste("No '", full_construct_display_name_grp_plot, "' score columns found.\n"))
}
  }
} else { cat("Warning: 'participantID' column not found. Skipping grouped boxplots.\n")}


cat("\n\n--- 6.3 Generating Combined Faceted Boxplot for All Constructs (with New Labels) ---
\n")
if ("participantID" %in% colnames(data) && length(existing_composite_score_columns) > 0) {
  data_long_all_constructs_viz <- data %>%
    select(participantID, all_of(existing_composite_score_columns)) %>%
    pivot_longer(cols = all_of(existing_composite_score_columns), names_to = "Score_Name",
values_to = "ScoreValue") %>%
    mutate(
      Short_Construct_Name = str_extract(Score_Name, "^(Anthro|Like|Intel|Trust)"),
      Robot = str_extract(Score_Name, "(Carl|Ryan|Ivan)"),
      Robot = factor(Robot, levels = robots_order_original_faceted), # Use new order

      # Create new column for plot labels based on the original Robot column
```

```
      Robot_Plot_Label = recode(Robot, "Ryan" = "Joint", "Ivan" = "Disjoint", "Carl" =
"Control"),
      # Factor the new column with the new display names for correct ordering in plots
      Robot_Plot_Label = factor(Robot_Plot_Label, levels = robot_display_names_faceted),

      Construct_Display = recode(Short_Construct_Name, !!!construct_name_map),
      Construct_Display = factor(Construct_Display, levels = unname(construct_name_map))
    ) %>%
    filter(!is.na(Robot) & !is.na(Construct_Display) & !is.na(ScoreValue))

  if (nrow(data_long_all_constructs_viz) > 0) {
    combined_faceted_plot_final <- ggplot(data_long_all_constructs_viz, aes(x =
Robot_Plot_Label, y = ScoreValue, fill = Robot_Plot_Label)) +
      geom_boxplot(outlier.colour = "red", outlier.shape = 16, outlier.size = 1.5, width = 0.7) +
      scale_fill_manual(values = robot_colors_faceted) + # Use the new color palette
      facet_wrap(~Construct_Display, scales = "free", ncol = 2) +
      labs(title = "Comparison of Subjective Ratings by Robot Condition", x = "Robot Condition",
y = "Mean Score") +
      theme_minimal(base_size = 12) +
      theme(legend.position = "none", strip.text = element_text(face="bold",size=11),
axis.text.x=element_text(angle=45,hjust=1,size=10), axis.title=element_text(size=11),
plot.title=element_text(hjust=0.5,size=14,face="bold"), panel.spacing=unit(1.5,"lines"))
    print(combined_faceted_plot_final)
  } else { cat("No data for combined faceted plot.\n")}
} else { cat("Warning: 'participantID' or composite scores missing. Skipping combined faceted
boxplot.\n")}


# --- 6.4 [NEW] Generating Combined Faceted Bar Chart with 95% Confidence Intervals ---
cat("\n\n--- 6.4 Generating Combined Bar Chart with 95% Confidence Intervals ---\n")
if (exists("data_long_all_constructs_viz") && nrow(data_long_all_constructs_viz) > 0) {

  # Group by the new Robot_Plot_Label column to calculate stats
  summary_stats_for_plot <- data_long_all_constructs_viz %>%
    group_by(Construct_Display, Robot_Plot_Label) %>%
    summarise(
      Mean = mean(ScoreValue, na.rm = TRUE),
      SD = sd(ScoreValue, na.rm = TRUE),
      N = n(),
      .groups = 'drop'
    ) %>%
    mutate(
      SE = SD / sqrt(N),
      CI_lower = Mean - 1.96 * SE,
      CI_upper = Mean + 1.96 * SE
    )
```

```
  # Use Robot_Plot_Label for x and fill aesthetics
  combined_faceted_barchart <- ggplot(summary_stats_for_plot, aes(x = Robot_Plot_Label, y =
Mean, fill = Robot_Plot_Label)) +
    geom_bar(stat = "identity", color = "black", width = 0.8) +
    geom_errorbar(
      aes(ymin = CI_lower, ymax = CI_upper),
      width = 0.25,
      linewidth = 0.5,
      color = "black"
    ) +
    geom_text(
      aes(label = sprintf("M = %.2f", Mean)),
      vjust = -2.5,
      color = "black",
      size = 3.5
    ) +
    facet_wrap(~Construct_Display, scales = "free", ncol = 2) +
    labs(
      title = "Mean Subjective Ratings by Robot Condition",
      subtitle = "Error bars represent 95% Confidence Intervals",
      x = "Robot Condition",
      y = "Mean Score"
    ) +
    scale_fill_manual(values = robot_colors_faceted) + # Use the new color palette
    scale_y_continuous(expand = expansion(mult = c(0, .15))) + # Give space for text labels
    theme_minimal(base_size = 12) +
    theme(
      legend.position = "none",
      strip.text = element_text(face = "bold", size = 11),
      axis.text.x = element_text(angle = 45, hjust = 1, size = 10),
      axis.title = element_text(size = 11),
      plot.title = element_text(hjust = 0.5, size = 14, face = "bold"),
      plot.subtitle = element_text(hjust = 0.5, size = 10),
      panel.spacing = unit(1.5, "lines"),
      panel.grid.major.x = element_blank() # Clean up grid lines
    )

  print(combined_faceted_barchart)
  cat("\n--- Bar chart with CIs generated successfully. ---\n")

} else {
  cat("Warning: Could not generate bar chart because the initial data processing step (6.3) failed
to produce data.\n")
}
```

```
# --- 7. IDENTIFY POTENTIAL OUTLIERS (1.5 * IQR Rule) ---
cat("\n\n--- 7. Identifying Potential Outliers for Composite Scores ---\n")
if (length(existing_composite_score_columns) > 0 && "participantID" %in% colnames(data)) {
  for (score_col_outlier_check in existing_composite_score_columns) {
    cat(paste0("\nChecking outliers for: ", score_col_outlier_check, "\n"))
    scores_vector_check <-
data[[score_col_outlier_check]][!is.na(data[[score_col_outlier_check]])]
    if (length(scores_vector_check) < 5) { cat("Not enough data.\n"); next }
    Q1_check <- quantile(scores_vector_check, 0.25); Q3_check <-
quantile(scores_vector_check, 0.75); IQR_val_check <- Q3_check - Q1_check
    lower_b_check <- Q1_check - 1.5 * IQR_val_check; upper_b_check <- Q3_check + 1.5 *
IQR_val_check
    potential_outliers_found <- data %>%
      filter((!!sym(score_col_outlier_check) < lower_b_check | !!sym(score_col_outlier_check) >
upper_b_check) & !is.na(!!sym(score_col_outlier_check))) %>%
      select(participantID, !!sym(score_col_outlier_check))
    if (nrow(potential_outliers_found) > 0) { cat("Potential outliers:\n");
print(potential_outliers_found) } else { cat("No outliers found.\n") }
  }
}
cat("\n--- Outlier Identification Complete. ---\n")

# --- 8. CHECK NORMALITY FOR EACH COMPOSITE SCORE ---
cat("\n\n--- 8. Checking Normality for Composite Scores ---\n")
for (score_col_norm_final in existing_composite_score_columns) {
  short_construct_final <- str_extract(score_col_norm_final, "^(Anthro|Like|Intel|Trust)")
  robot_name_final <- str_extract(score_col_norm_final, "(Carl|Ryan|Ivan)")
  full_construct_final <- construct_name_map[[short_construct_final]]
  plot_title_hist_final <- paste("Hist & Density:", full_construct_final, "-", robot_name_final)
  if (is.na(full_construct_final)) plot_title_hist_final <- paste("Hist & Density:",
score_col_norm_final)

  cat(paste0("\n--- Normality Check for: ", score_col_norm_final, " ---\n"))
  score_values_for_norm <- data[[score_col_norm_final]][!is.na(data[[score_col_norm_final]])]
  if (length(score_values_for_norm) >= 3) {
    hist_plot_final_render <- ggplot(data, aes(x = .data[[score_col_norm_final]])) +
      geom_histogram(aes(y=after_stat(density)), binwidth = 0.5, fill="cornflowerblue",
color="black", alpha=0.7, na.rm=TRUE) +
      geom_density(alpha = 0.5, fill="darkorange", colour="darkorange", na.rm=TRUE) +
      labs(title=plot_title_hist_final, x = "Score", y = "Density") + theme_minimal()
    print(hist_plot_final_render)

    qq_plot_title_final <- paste("Q-Q Plot:", full_construct_final, "-", robot_name_final)
    if (is.na(full_construct_final)) qq_plot_title_final <- paste("Q-Q Plot:", score_col_norm_final)
```

```
    qq_plot_final_render <- ggqqplot(data, x=score_col_norm_final, conf.int = TRUE, ggtheme =
theme_minimal(), ylab="SampleQ", xlab="TheoreticalQ", title=qq_plot_title_final)
    print(qq_plot_final_render)

    if (length(score_values_for_norm) <= 5000) {
      shapiro_test_final <- shapiro.test(score_values_for_norm)
      cat(paste0("Shapiro-Wilk for ", score_col_norm_final, ":
W=",round(shapiro_test_final$statistic,3),", p=",round(shapiro_test_final$p.value,3),"\n"))
    } else { cat("N > 5000, Shapiro-Wilk may not be optimal.\n") }
  } else { cat("Not enough data for normality check.\n")}
}
cat("\n--- Normality Assessment Complete. ---\n")

# --- 9. CRONBACH'S ALPHA FOR SCALE RELIABILITY ---
cat("\n\n--- 9. Calculating Cronbach's Alpha ---\n")
all_item_sets_orig_alpha_final <- list(
  Anthro_Carl = anthro_carl_items, Like_Carl = like_carl_items, Intel_Carl = intel_carl_items,
Trust_Carl = trust_carl_items,
  Anthro_Ryan = anthro_ryan_items, Like_Ryan = like_ryan_items, Intel_Ryan =
intel_ryan_items, Trust_Ryan = trust_ryan_items,
  Anthro_Ivan = anthro_ivan_items, Like_Ivan = like_ivan_items, Intel_Ivan = intel_ivan_items,
Trust_Ivan = trust_ivan_items
)
alpha_results_list_full <- list()
for (scale_name_alpha_run in names(all_item_sets_orig_alpha_final)) {
  items_for_alpha_run <- all_item_sets_orig_alpha_final[[scale_name_alpha_run]]
  present_items_for_alpha_run <- items_for_alpha_run[items_for_alpha_run %in%
colnames(data)]
  if (length(present_items_for_alpha_run) >= 2) {
    data_subset_for_alpha_run <- data %>% select(all_of(present_items_for_alpha_run)) %>%
mutate(across(everything(), as.numeric))
    data_subset_complete_for_alpha_run <-
data_subset_for_alpha_run[rowSums(is.na(data_subset_for_alpha_run)) <
ncol(data_subset_for_alpha_run), ] # Keep rows with at least one non-NA value
    if(nrow(data_subset_complete_for_alpha_run) >=2 &&
ncol(data_subset_complete_for_alpha_run) >=2) { # Check if still valid after NA row removal
      alpha_obj_run_final <- psych::alpha(data_subset_complete_for_alpha_run,
check.keys=TRUE, use="pairwise.complete.obs") # pairwise for robustness
      alpha_results_list_full[[scale_name_alpha_run]] <- alpha_obj_run_final$total
    } else { alpha_results_list_full[[scale_name_alpha_run]] <- list(std.alpha = NA_real_) }
  } else { alpha_results_list_full[[scale_name_alpha_run]] <- list(std.alpha = NA_real_) }
}
alpha_summary_df_to_show <- tibble(Scale=names(alpha_results_list_full),
Std_Alpha=sapply(alpha_results_list_full, function(x) if(is.list(x) && "std.alpha" %in%
names(x)) round(x$std.alpha,3) else NA_real_))
print(alpha_summary_df_to_show, n=Inf)
```

```
cat("\n--- Cronbach's Alpha calculation finished. ---\n")

# --- 10. PARAMETRIC TESTING - REPEATED MEASURES ANOVA ---
cat("\n\n--- 10. Parametric Testing (Repeated Measures ANOVAs) ---\n")
if (!"participantID" %in% colnames(data)) { stop("Error: 'participantID' column required for
ANOVA.") }

for (construct_short_final_anova in constructs_short_names_for_iteration) {
  full_construct_final_anova <- construct_name_map[[construct_short_final_anova]]
  cat(paste0("\n\n--- RM ANOVA for: ", full_construct_final_anova, " Scores ---\n"))

  # IMPORTANT: This section uses the original `robots_order` to find the correct columns
  composite_cols_final_anova <- paste0(construct_short_final_anova, "_", robots_order,
"_Score")
  existing_cols_final_anova <- intersect(composite_cols_final_anova, colnames(data))
  if (length(existing_cols_final_anova) != length(robots_order)) { cat(paste0("Skipping ",
full_construct_final_anova, ": not all score columns found.\n")); next }

  data_long_final_anova <- data %>% select(participantID, all_of(existing_cols_final_anova))
%>%
    pivot_longer(cols = all_of(existing_cols_final_anova), names_to = "Robot_Condition_Raw",
values_to = "ScoreValue") %>%
    mutate(Robot = str_extract(Robot_Condition_Raw, paste(robots_order, collapse="|")), Robot
= factor(Robot, levels = robots_order))

  data_long_complete_final_anova <- data_long_final_anova %>% filter(!is.na(ScoreValue))
  if(nrow(data_long_complete_final_anova) == 0) { cat(paste0("Skipping ",
full_construct_final_anova, ": no data after NA removal.\n")); next }

  n_distinct_robots_final_anova <-
n_distinct(data_long_complete_final_anova$Robot[!is.na(data_long_complete_final_anova$Rob
ot)])
  if (n_distinct_robots_final_anova < length(robots_order)) { cat(paste0("Skipping ",
full_construct_final_anova, ": not all robots present.\n")); next }

  subject_counts_final_anova <- data_long_complete_final_anova %>% group_by(participantID)
%>% summarise(n_cond_answered = n_distinct(Robot), .groups = 'drop')
  complete_subjects_final_anova <- subject_counts_final_anova %>% filter(n_cond_answered
== n_distinct_robots_final_anova) %>% pull(participantID)
  if (length(complete_subjects_final_anova) < 2) { cat(paste0("Skipping ",
full_construct_final_anova, ": <2 subjects with complete data.\n")); next }

  data_anova_for_rstatix <- data_long_complete_final_anova %>% filter(participantID %in%
complete_subjects_final_anova)
```

```
  cat(paste0("RM ANOVA for '", full_construct_final_anova, "' (N=",
length(complete_subjects_final_anova), " subjects)...\n"))
  tryCatch({
    res_aov_actual <- rstatix::anova_test(data = data_anova_for_rstatix, dv = ScoreValue, wid =
participantID, within = Robot, effect.size = "ges" )
    print(res_aov_actual)

    p_val_aov_actual <- NA; anova_table_from_rstatix <- NULL
    if(is.list(res_aov_actual) && "ANOVA" %in% names(res_aov_actual)) {
anova_table_from_rstatix <- res_aov_actual$ANOVA }
    else if(is.data.frame(res_aov_actual)) { anova_table_from_rstatix <- res_aov_actual }
    if(!is.null(anova_table_from_rstatix) && "p" %in% colnames(anova_table_from_rstatix) &&
"Effect" %in% colnames(anova_table_from_rstatix)) {
      p_row_val_actual <- anova_table_from_rstatix[anova_table_from_rstatix$Effect ==
"Robot",]; if(nrow(p_row_val_actual) == 1) p_val_aov_actual <- p_row_val_actual$p
    }
    if (!is.na(p_val_aov_actual) && p_val_aov_actual < 0.05) {
      cat(paste0("ANOVA for '", full_construct_final_anova, "' significant. Pairwise
(Bonferroni):\n"))
      print(data_anova_for_rstatix %>% rstatix::pairwise_t_test(ScoreValue ~ Robot, paired =
TRUE, p.adjust.method = "bonferroni"))
    } else { cat(paste0("ANOVA for '", full_construct_final_anova, "' NOT significant or p-value
not extracted.\n")) }
  }, error = function(e) { cat(paste0("ERROR RM ANOVA for '", full_construct_final_anova, "':
", e$message, "\n")) })
}
cat("\n--- Parametric testing finished. ---\n")

# --- 11. SAVE FINAL DATASET ---
final_output_filename <- "qualtrics_data_fully_processed_with_all_analyses.csv"
cat(paste0("\n\n--- 11. Saving final data to '", final_output_filename, "' ---\n"))
write_csv(data, final_output_filename)
cat("--- Script finished. Final data saved. ---\n")
```

Python-Script for generating Heatmaps

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# --- USER CONFIGURATION ---
FULL_DATA_FILE = "newest_combined_eyetracking_data.csv"
BACKGROUND_IMAGE_FILE = "carl.png"

def generate_heatmap(data_filepath, image_filepath, target_robot, target_difficulty):
    """
    Generates a high-visibility heatmap with a shorter, closer colorbar legend.
    """
    print(f"Generating heatmap for: {target_robot} / {target_difficulty}...")

    # --- 1. Load and Prepare Data ---
    try:
        df = pd.read_csv(data_filepath, decimal=',')
        bg_img = mpimg.imread(image_filepath)
        img_height, img_width, _ = bg_img.shape
    except FileNotFoundError as e:
        print(f"FATAL ERROR: Could not find a required file. {e}")
        return

    coord_cols = ['Fixation point X (MCSnorm)', 'Fixation point Y (MCSnorm)']
    for col in coord_cols:
        df[col] = pd.to_numeric(df[col], errors='coerce')

    condition_df = df[
        (df['Robot'] == target_robot) &
        (df['difficulty'] == target_difficulty) &
        (df['Eye movement type'] == 'Fixation')
    ].dropna(subset=coord_cols).copy()

    if condition_df.empty:
        print("No fixation data found for the selected condition.")
        return

    print(f"Found {len(condition_df)} fixations for this condition.")

    condition_df['x_pixel'] = condition_df['Fixation point X (MCSnorm)'] * img_width
    condition_df['y_pixel'] = condition_df['Fixation point Y (MCSnorm)'] * img_height

    # --- 2. Create the Heatmap Plot ---
```

```
    fig, ax = plt.subplots(figsize=(12, 9))

    # Display the background image
    ax.imshow(bg_img)

    # --- UPDATED: Final adjustments for legend size and position ---
    sns.kdeplot(
        x=condition_df['x_pixel'],
        y=condition_df['y_pixel'],
        ax=ax,
        fill=True,
        cmap="rocket_r",
        alpha=0.75,
        thresh=0.05,
        bw_adjust=0.8,
        cbar=True,
        cbar_kws={
            'label': 'Fixation Density',
            'shrink': 0.4,    # --- REDUCED: Makes the colorbar even shorter (40% of plot height) ---
            'pad': 0.02       # --- ADDED: Moves the colorbar closer to the plot ---
        }
    )

    ax.set_title(f"Fixation Heatmap for: {target_robot} / {target_difficulty}", fontsize=16)
    ax.axis('off')

    plt.tight_layout()
    plt.show()


# --- Main Execution Block (Set to compare Carl: Easy vs. Hard) ---
if __name__ == "__main__":
    print("--- Generating heatmap for carl / easy ---")
    generate_heatmap(data_filepath=FULL_DATA_FILE,
                     image_filepath=BACKGROUND_IMAGE_FILE,
                     target_robot='Carl condition',
                     target_difficulty='easy')

    print("\n--- Generating a second heatmap for comparison ---")
    generate_heatmap(data_filepath=FULL_DATA_FILE,
                     image_filepath=BACKGROUND_IMAGE_FILE,
                     target_robot='Carl condition',
                     target_difficulty='hard')
```

Python Script for Analysis of Dwell Time

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pingouin as pg
import os

# --- USER CONFIGURATION ---
FULL_DATA_FILE = "newest_combined_eyetracking_data.csv"
# The directory where results will be saved
OUTPUT_DIR = "analysis_results"

# Outlier threshold
SD_THRESHOLD = 2.5

def run_analysis_for_aoi(df, target_aoi_col, target_aoi_name, sd_thresh):
    """
    Calculates Proportional Dwell Time for a given AOI, removes outliers,
    runs a 2x3 repeated-measures ANOVA, and generates visualizations.
    """
    print("\n" + "="*80)
    print(f"Running Analysis for AOI: '{target_aoi_name}' (Column: {target_aoi_col})")
    print("="*80)

    # --- 1. Calculate Proportional Dwell Time ---
    print("\n[Step 1] Calculating Proportional Dwell Time...")
    # Isolate only fixation events
    fixations_df = df[df['Eye movement type'] == 'Fixation'].copy()
    # Filter for fixations on the current target AOI
    aoi_fixations = fixations_df[fixations_df[target_aoi_col] == True]
    # Group by trial and SUM the DURATION of fixations for the AOI
    dwell_times = aoi_fixations.groupby(['ParticipantID',
'classification_timeframe_number'])['Eye movement event
duration'].sum().to_frame(name='Dwell_Time_ms').reset_index()

    # Calculate Total Trial Duration from the main df
    total_trial_durations = df.groupby(['ParticipantID', 'classification_timeframe_number'])['Eye
movement event duration'].sum().to_frame(name='Total_Trial_Duration_ms').reset_index()

    # Create a complete list of all trials to merge onto
    all_trials = df[['ParticipantID', 'classification_timeframe_number', 'Robot',
'difficulty']].drop_duplicates()
    # Merge AOI dwell times and total trial durations
```

```
    analysis_df = pd.merge(all_trials, dwell_times, on=['ParticipantID',
'classification_timeframe_number'], how='left')
    analysis_df = pd.merge(analysis_df, total_trial_durations, on=['ParticipantID',
'classification_timeframe_number'], how='left')

    # Fill NaNs and calculate the proportion
    analysis_df['Dwell_Time_ms'].fillna(0, inplace=True)
    analysis_df['Proportional_Dwell_Time'] = np.where(analysis_df['Total_Trial_Duration_ms'] >
0,
                                            analysis_df['Dwell_Time_ms'] /
analysis_df['Total_Trial_Duration_ms'],
                                            0)
    print(f"Proportional Dwell Time calculated for {len(analysis_df)} trials.")

    # --- 2. Outlier Removal ---
    print(f"\n[Step 2] Checking for outliers in 'Proportional_Dwell_Time' for
'{target_aoi_name}'...")
    original_rows = len(analysis_df)
    def remove_outliers_by_sd(df, group_cols, value_col, threshold):
        def remove_group_outliers(group):
            mean = group[value_col].mean()
            std_dev = group[value_col].std()
            if pd.isna(std_dev) or std_dev == 0: return group
            lower_bound = mean - threshold * std_dev
            upper_bound = mean + threshold * std_dev
            return group[(group[value_col] >= lower_bound) & (group[value_col] <= upper_bound)]
        return df.groupby(group_cols, group_keys=False).apply(remove_group_outliers)

    analysis_df = remove_outliers_by_sd(analysis_df,
                                group_cols=['Robot', 'difficulty', 'ParticipantID'],
                                value_col='Proportional_Dwell_Time',
                                threshold=sd_thresh)
    outliers_removed = original_rows - len(analysis_df)
    percentage_lost = (outliers_removed / original_rows) * 100 if original_rows > 0 else 0
    print(f" Removed {outliers_removed} outlier(s) ({percentage_lost:.2f}% of the data).")

    # Save the cleaned data to a unique file
    output_filename = f"proportional_dwell_time_{target_aoi_name.replace(' ',
'_').lower()}_results.csv"
    output_filepath = os.path.join(OUTPUT_DIR, output_filename)
    analysis_df.to_csv(output_filepath, index=False)
    print(f"Cleaned results for '{target_aoi_name}' saved to '{output_filepath}'.")

    # --- 3. Visualize and Analyze ---
    print(f"\n[Step 3] Visualizing and running ANOVA for '{target_aoi_name}'...")
```

```
    analysis_df['Proportional_Dwell_Time_Percent'] = analysis_df['Proportional_Dwell_Time'] *
100.0

    robot_name_map = {
        "Ryan condition": "Joint condition",
        "Ivan condition": "Disjoint condition",
        "Carl condition": "Control condition"
    }
    analysis_df['Robot'] = analysis_df['Robot'].map(robot_name_map)
    robot_order = ["Joint condition", "Disjoint condition", "Control condition"]
    if all(robot in analysis_df['Robot'].unique() for robot in robot_order):
        analysis_df['Robot'] = pd.Categorical(analysis_df['Robot'], categories=robot_order,
ordered=True)

    # Create grouped bar chart
    plt.figure(figsize=(12, 8))
    sns.barplot(x='Robot', y='Proportional_Dwell_Time_Percent', hue='difficulty',
data=analysis_df, palette="viridis", capsize=.05, errorbar="se")
    plt.title(f"Mean Proportional Dwell Time on {target_aoi_name}\nby Condition and
Difficulty")
    plt.ylabel('Mean Proportional Dwell Time (%)')
    plt.xlabel('Robotic Condition')
    plt.legend(title='Difficulty')
    # Save the plot to a file
    plot_filename = f"plot_{target_aoi_name.replace(' ', '_').lower()}.png"
    plot_filepath = os.path.join(OUTPUT_DIR, plot_filename)
    plt.savefig(plot_filepath)
    plt.show()
    print(f"Plot for '{target_aoi_name}' saved to '{plot_filepath}'.")


    # Perform ANOVA
    aov = pg.rm_anova(data=analysis_df,
                      dv='Proportional_Dwell_Time',
                      within=['Robot', 'difficulty'],
                      subject='ParticipantID',
                      detailed=True)
    print(f"\n--- ANOVA Results for Proportional Dwell Time on '{target_aoi_name}' ---")
    pg.print_table(aov)

    # Conditional Post-Hoc tests
    is_robot_sig = aov.loc[aov['Source'] == 'Robot', 'p-unc'].iloc[0] < 0.05
    is_interaction_sig = aov.loc[aov['Source'] == 'Robot * difficulty', 'p-unc'].iloc[0] < 0.05
    if is_robot_sig or is_interaction_sig:
        print(f"\n--- Post-Hoc Tests for Proportional Dwell Time on '{target_aoi_name}' ---")
```

```
        posthocs = pg.pairwise_tests(data=analysis_df, dv='Proportional_Dwell_Time',
within=['Robot', 'difficulty'], subject='ParticipantID', padjust='bonf')
        print(posthocs)


# --- Main Execution Block ---
if __name__ == "__main__":
    # --- 1. Load and Prepare Data ONCE ---
    print("Loading and preparing main data file...")
    try:
        main_df = pd.read_csv(FULL_DATA_FILE, decimal=',')
    except FileNotFoundError:
        print(f"FATAL ERROR: Could not find '{FULL_DATA_FILE}'")
        exit() # Use exit() in main block

    # Create the output directory if it doesn't exist
    if not os.path.exists(OUTPUT_DIR):
        os.makedirs(OUTPUT_DIR)

    # Standard data cleaning
    main_df['classification_timeframe_number'] =
pd.to_numeric(main_df['classification_timeframe_number'], errors='coerce')
    main_df['Eye movement event duration'] = pd.to_numeric(main_df['Eye movement event
duration'], errors='coerce')
    main_df.dropna(subset=['classification_timeframe_number', 'Eye movement event duration'],
inplace=True)
    main_df['classification_timeframe_number'] =
main_df['classification_timeframe_number'].astype('Int64')
    if 'ParticipantID' not in main_df.columns: main_df['ParticipantID'] = 'Unknown'
    else: main_df['ParticipantID'] = main_df['ParticipantID'].ffill().bfill()
    grouping_cols_for_ffill = ['ParticipantID', 'classification_timeframe_number']
    cols_to_ffill = ['Robot', 'difficulty']
    for col_ffill in cols_to_ffill:
        if col_ffill in main_df.columns:
            main_df[col_ffill] = main_df.groupby(grouping_cols_for_ffill,
group_keys=False)[col_ffill].ffill().bfill()
    print("Data loaded and prepared.")

    # --- 2. Create Combined AOI Column ---
    # The | operator works as a boolean OR for pandas columns.
    print("\nCreating combined 'Classification Buttons' AOI...")
    main_df['classification_buttons'] = main_df['is_true_category'] | main_df['is_false_category']
    print("Combined AOI created.")


    # --- 3. Define AOIs and Run Analysis for Each ---
```

```
    aois_to_analyze = [
        {'col': 'is_face', 'name': 'Robot Face'},
        {'col': 'is_cards', 'name': 'Cards'},
        {'col': 'classification_buttons', 'name': 'Classification Buttons'}
    ]

    for aoi in aois_to_analyze:
        run_analysis_for_aoi(df=main_df.copy(), # Pass a copy to ensure original df is unchanged
                             target_aoi_col=aoi['col'],
                             target_aoi_name=aoi['name'],
                             sd_thresh=SD_THRESHOLD)

    print("\nAll analyses completed")
```

Python Script for AOI Frequency Analysis

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import pingouin as pg
import os

# --- USER CONFIGURATION ---
FULL_DATA_FILE = "newest_combined_eyetracking_data.csv"
# The directory where results will be saved
OUTPUT_DIR = "analysis_results_fixation_count"

# Outlier threshold
SD_THRESHOLD = 2.5

def run_proportional_fixation_analysis(df, target_aoi_col, target_aoi_name, sd_thresh):
    """
    Calculates Proportional Fixation Count for a given AOI, removes outliers,
    runs a 2x3 repeated-measures ANOVA, and generates visualizations.
    """
    print("\n" + "="*80)
    print(f"Running Analysis for AOI: '{target_aoi_name}' (Column: {target_aoi_col})")
    print("="*80)

    # --- 1. Calculate Proportional Fixation Count ---
    print("\n[Step 1] Calculating Proportional Fixation Count...")

    # Isolate only fixation events from the main dataframe
    fixations_df = df[df['Eye movement type'] == 'Fixation'].copy()
```

```
    # Filter for fixations on the current target AOI
    aoi_fixations = fixations_df[fixations_df[target_aoi_col] == True]

    # Group by trial and COUNT the fixations for the AOI
    aoi_fix_counts = aoi_fixations.groupby(['ParticipantID',
'classification_timeframe_number']).size().to_frame(name='AOI_Fixation_Count').reset_index()

    # Calculate TOTAL number of fixations for each trial
    total_fix_counts = fixations_df.groupby(['ParticipantID',
'classification_timeframe_number']).size().to_frame(name='Total_Trial_Fixation_Count').reset_i
ndex()

    # Create a complete list of all trials to merge onto
    all_trials = df[['ParticipantID', 'classification_timeframe_number', 'Robot',
'difficulty']].drop_duplicates()

    # Merge AOI counts and total trial counts
    analysis_df = pd.merge(all_trials, aoi_fix_counts, on=['ParticipantID',
'classification_timeframe_number'], how='left')
    analysis_df = pd.merge(analysis_df, total_fix_counts, on=['ParticipantID',
'classification_timeframe_number'], how='left')

    # Fill NaNs and calculate the proportion
    analysis_df['AOI_Fixation_Count'].fillna(0, inplace=True)
    analysis_df['Total_Trial_Fixation_Count'].fillna(0, inplace=True) # A trial might have no
fixations at all
    analysis_df['Proportional_Fixation_Count'] =
np.where(analysis_df['Total_Trial_Fixation_Count'] > 0,
                                            analysis_df['AOI_Fixation_Count'] /
analysis_df['Total_Trial_Fixation_Count'],
                                            0)
    print(f"Proportional Fixation Count calculated for {len(analysis_df)} trials.")

    # --- 2. Outlier Removal ---
    print(f"\n[Step 2] Checking for outliers in 'Proportional_Fixation_Count' for
'{target_aoi_name}'...")
    original_rows = len(analysis_df)
    def remove_outliers_by_sd(df, group_cols, value_col, threshold):
        def remove_group_outliers(group):
            mean = group[value_col].mean()
            std_dev = group[value_col].std()
            if pd.isna(std_dev) or std_dev == 0: return group
            lower_bound = mean - threshold * std_dev
            upper_bound = mean + threshold * std_dev
            return group[(group[value_col] >= lower_bound) & (group[value_col] <= upper_bound)]
```

```
        return df.groupby(group_cols, group_keys=False).apply(remove_group_outliers)

    analysis_df = remove_outliers_by_sd(analysis_df,
                                        group_cols=['Robot', 'difficulty', 'ParticipantID'],
                                        value_col='Proportional_Fixation_Count',
                                        threshold=sd_thresh)
    outliers_removed = original_rows - len(analysis_df)
    percentage_lost = (outliers_removed / original_rows) * 100 if original_rows > 0 else 0
    print(f" Removed {outliers_removed} outlier(s) ({percentage_lost:.2f}% of the data).")

    # Save the cleaned data to a unique file
    output_filename = f"proportional_fixation_count_{target_aoi_name.replace(' ',
'_').lower()}_results.csv"
    output_filepath = os.path.join(OUTPUT_DIR, output_filename)
    analysis_df.to_csv(output_filepath, index=False)
    print(f"Cleaned results for '{target_aoi_name}' saved to '{output_filepath}'.")

    # --- 3. Visualize and Analyze ---
    print(f"\n[Step 3] Visualizing and running ANOVA for '{target_aoi_name}'...")
    analysis_df['Proportional_Fixation_Count_Percent'] =
analysis_df['Proportional_Fixation_Count'] * 100.0

    robot_name_map = {
        "Ryan condition": "Joint condition",
        "Ivan condition": "Disjoint condition",
        "Carl condition": "Control condition"
    }
    analysis_df['Robot'] = analysis_df['Robot'].map(robot_name_map)
    robot_order = ["Joint condition", "Disjoint condition", "Control condition"]
    if all(robot in analysis_df['Robot'].unique() for robot in robot_order):
        analysis_df['Robot'] = pd.Categorical(analysis_df['Robot'], categories=robot_order,
ordered=True)

    # Create grouped bar chart
    plt.figure(figsize=(12, 8))
    sns.barplot(x='Robot', y='Proportional_Fixation_Count_Percent', hue='difficulty',
data=analysis_df, palette="magma", capsize=.05, errorbar="se")
    plt.title(f"Mean Proportional Fixation Count on {target_aoi_name}\nby Condition and
Difficulty")
    plt.ylabel('Mean Proportional Fixation Count (%)')
    plt.xlabel('Robotic Condition')
    plt.legend(title='Difficulty')
    # Save the plot to a file
    plot_filename = f"plot_fixation_count_{target_aoi_name.replace(' ', '_').lower()}.png"
    plot_filepath = os.path.join(OUTPUT_DIR, plot_filename)
    plt.savefig(plot_filepath)
```

```
    plt.show()
    print(f"Plot for '{target_aoi_name}' saved to '{plot_filepath}'.")

    # Perform ANOVA
    aov = pg.rm_anova(data=analysis_df,
                      dv='Proportional_Fixation_Count',
                      within=['Robot', 'difficulty'],
                      subject='ParticipantID',
                      detailed=True)
    print(f"\n--- ANOVA Results for Proportional Fixation Count on '{target_aoi_name}' ---")
    pg.print_table(aov)

    # Conditional Post-Hoc tests
    is_robot_sig = aov.loc[aov['Source'] == 'Robot', 'p-unc'].iloc[0] < 0.05
    is_interaction_sig = aov.loc[aov['Source'] == 'Robot * difficulty', 'p-unc'].iloc[0] < 0.05
    if is_robot_sig or is_interaction_sig:
        print(f"\n--- Post-Hoc Tests for Proportional Fixation Count on '{target_aoi_name}' ---")
        posthocs = pg.pairwise_tests(data=analysis_df, dv='Proportional_Fixation_Count',
within=['Robot', 'difficulty'], subject='ParticipantID', padjust='bonf')
        print(posthocs)

# --- Main Execution Block ---
if __name__ == "__main__":
    # --- 1. Load and Prepare Data ONCE ---
    print("Loading and preparing main data file...")
    try:
        main_df = pd.read_csv(FULL_DATA_FILE, decimal=',')
    except FileNotFoundError:
        print(f"FATAL ERROR: Could not find '{FULL_DATA_FILE}'")
        exit()

    # Create the output directory if it doesn't exist
    if not os.path.exists(OUTPUT_DIR):
        os.makedirs(OUTPUT_DIR)

    # Standard data cleaning
    main_df['classification_timeframe_number'] =
pd.to_numeric(main_df['classification_timeframe_number'], errors='coerce')
    main_df.dropna(subset=['classification_timeframe_number'], inplace=True)
    main_df['classification_timeframe_number'] =
main_df['classification_timeframe_number'].astype('Int64')
    if 'ParticipantID' not in main_df.columns: main_df['ParticipantID'] = 'Unknown'
    else: main_df['ParticipantID'] = main_df['ParticipantID'].ffill().bfill()
    grouping_cols_for_ffill = ['ParticipantID', 'classification_timeframe_number']
    cols_to_ffill = ['Robot', 'difficulty']
    for col_ffill in cols_to_ffill:
```

```
        if col_ffill in main_df.columns:
            main_df[col_ffill] = main_df.groupby(grouping_cols_for_ffill,
group_keys=False)[col_ffill].ffill().bfill()
    print("Data loaded and prepared.")

    # --- 2. Create Combined AOI Column ---
    print("\nCreating combined 'Classification Buttons' AOI...")
    main_df['classification_buttons'] = main_df['is_true_category'] | main_df['is_false_category']
    print("Combined AOI created.")

    # --- 3. Define AOIs and Run Analysis for Each ---
    aois_to_analyze = [
        {'col': 'is_face', 'name': 'Robot Face'},
        {'col': 'is_cards', 'name': 'Cards'},
        {'col': 'classification_buttons', 'name': 'Classification Buttons'}
    ]

    for aoi in aois_to_analyze:
        run_proportional_fixation_analysis(df=main_df.copy(),
                                           target_aoi_col=aoi['col'],
                                           target_aoi_name=aoi['name'],
                                           sd_thresh=SD_THRESHOLD)

    print("\nAll fixation count analyses complete.")
```

Python Script for Advancced AOI Transition Analysis

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency

# --- USER CONFIGURATION ---
FULL_DATA_FILE = "newest_combined_eyetracking_data.csv"

# Define your key AOIs and give them short names for the matrix
AOI_DEFINITIONS = {
    'Robot': 'is_robot',
    'Cards': 'is_cards',
    'Classification': 'classification_category'
}
```

```
def run_full_transition_analysis(data_filepath, aoi_defs):
    """
    Calculates effective AOI transitions, runs statistical tests (Chi-Squared)
    with post-hoc analysis, and then visualizes the results.
    """
    # --- 1. Load and Prepare Data ---
    print("[Step 1] Loading data and calculating all transitions...")
    try:
        df = pd.read_csv(data_filepath, decimal=',')
    except FileNotFoundError as e:
        print(f"FATAL ERROR: Could not find the data file. {e}")
        return

    # Standard data cleaning
    df.dropna(subset=['robot_appearance_timeframe_number', 'Robot', 'difficulty'], inplace=True)
    df['classification_category'] = (df.get('is_false_category', False) | df.get('is_true_category',
False))

    def get_aoi_state(row, aoi_definitions):
        for aoi_name, col_name in aoi_definitions.items():
            if col_name in row and row[col_name]:
                return aoi_name
        return 'Outside'
    df['aoi_state'] = df.apply(lambda row: get_aoi_state(row, aoi_defs), axis=1)

    # --- 2. Build a Master List of ALL Transitions ---
    all_transitions = []
    for name, group in df.groupby(['ParticipantID', 'robot_appearance_timeframe_number']):
        robot_condition = group['Robot'].iloc[0]
        difficulty_level = group['difficulty'].iloc[0]

        simplified_sequence = group['aoi_state'][group['aoi_state'].shift() != group['aoi_state']]
        effective_sequence = simplified_sequence[simplified_sequence != 'Outside']

        if len(effective_sequence) > 1:
            trial_transitions = list(zip(effective_sequence, effective_sequence.iloc[1:]))
            for trans_from, trans_to in trial_transitions:
                all_transitions.append({
                    'From': trans_from,
                    'To': trans_to,
                    'Robot': robot_condition,
                    'Difficulty': difficulty_level
                })

    if not all_transitions:
        print("No transitions were found.")
```

```
        return

    master_transition_df = pd.DataFrame(all_transitions)
    print("Master list of all transitions created successfully.")

    # --- 3. Perform Overall Statistical Tests (Chi-Squared) ---
    print("\n[Step 3] Performing Chi-Squared tests for overall significance...")

    # Test 1: Does the transition pattern depend on the Robot?
    print("\n--- Test 1: Do transition patterns differ by ROBOT? ---")
    # The crosstab function creates the contingency table of observed counts
    contingency_table_robot = pd.crosstab(master_transition_df['From'],
[master_transition_df['To'], master_transition_df['Robot']])
    chi2, p, dof, expected_robot = chi2_contingency(contingency_table_robot)
    print(f"Chi-Squared Statistic: {chi2:.2f}, p-value: {p:.4f}")
    if p < 0.05:
        print("Conclusion: YES, the pattern of transitions is significantly different across the robot
conditions.")
        # --- MODIFICATION START: POST-HOC FOR ROBOT CONDITION ---
        print("\n--- Post-Hoc Analysis: Standardized Residuals for Robot Condition ---")
        print("This shows which specific transitions occurred significantly more or less often than
expected for each robot.")
        # Rule of thumb: A residual > 1.96 or < -1.96 is significant at p < .05
        residuals_robot = (contingency_table_robot - expected_robot) / np.sqrt(expected_robot)

        # Flatten the table for easier parsing
        stacked_residuals_robot = residuals_robot.stack(level=[0, 1]).reset_index()
        stacked_residuals_robot.columns = ['From', 'To', 'Robot', 'Residual']

        # Filter for significant results
        significant_residuals_robot =
stacked_residuals_robot[np.abs(stacked_residuals_robot['Residual']) > 1.96]

        for index, row in significant_residuals_robot.sort_values(by='Residual',
ascending=False).iterrows():
            direction = "more" if row['Residual'] > 0 else "less"
            print(f"  - In '{row['Robot']}', transitions from '{row['From']}' to '{row['To']}' occurred
{direction} frequently than expected (Residual: {row['Residual']:.2f})")
        # --- MODIFICATION END ---
    else:
        print("Conclusion: NO, the pattern of transitions is not significantly different across the
robot conditions.")


    # Test 2: Does the transition pattern depend on Difficulty?
    print("\n--- Test 2: Do transition patterns differ by DIFFICULTY? ---")
```

```
    contingency_table_difficulty = pd.crosstab(master_transition_df['From'],
[master_transition_df['To'], master_transition_df['Difficulty']])
    chi2, p, dof, expected_difficulty = chi2_contingency(contingency_table_difficulty)
    print(f"Chi-Squared Statistic: {chi2:.2f}, p-value: {p:.4f}")
    if p < 0.05:
        print("Conclusion: YES, the pattern of transitions is significantly different between easy
and hard trials.")
        # --- MODIFICATION START: POST-HOC FOR DIFFICULTY ---
        print("\n--- Post-Hoc Analysis: Standardized Residuals for Difficulty ---")
        print("This shows which specific transitions occurred significantly more or less often than
expected for each difficulty level.")

        residuals_difficulty = (contingency_table_difficulty - expected_difficulty) /
np.sqrt(expected_difficulty)

        # Flatten the table for easier parsing
        stacked_residuals_difficulty = residuals_difficulty.stack(level=[0, 1]).reset_index()
        stacked_residuals_difficulty.columns = ['From', 'To', 'Difficulty', 'Residual']

        # Filter for significant results
        significant_residuals_difficulty =
stacked_residuals_difficulty[np.abs(stacked_residuals_difficulty['Residual']) > 1.96]

        for index, row in significant_residuals_difficulty.sort_values(by='Residual',
ascending=False).iterrows():
            direction = "more" if row['Residual'] > 0 else "less"
            print(f"  - In '{row['Difficulty']}' trials, transitions from '{row['From']}' to '{row['To']}'
occurred {direction} frequently than expected (Residual: {row['Residual']:.2f})")
        # --- MODIFICATION END ---
    else:
        print("Conclusion: NO, the pattern of transitions is not significantly different between easy
and hard trials.")


    # --- 4. Generate Descriptive Heatmaps for Each Condition ---
    print("\n[Step 4] Generating descriptive probability matrices and heatmaps for each
condition...")

    robot_name_map = {
        "Ryan condition": "Joint Condition (Ryan)",
        "Ivan condition": "Disjoint Condition (Ivan)",
        "Carl condition": "Control Condition (Carl)"
    }

    # Loop through each condition to generate its specific matrix and heatmap
    for difficulty in ['easy', 'hard']:
```

```
    print("\n" + "#"*30 + f"\n#   ANALYSIS FOR {difficulty.upper()} TRIALS   #\n" + "#"*30
+ "\n")
    for robot in ["Ryan condition", "Ivan condition", "Carl condition"]:
        print("\n" + "="*80)
        print(f"CONDITION: {robot} / {difficulty}")
        print("="*80)

        condition_subset_df = master_transition_df[
            (master_transition_df['Robot'] == robot) &
            (master_transition_df['Difficulty'] == difficulty)
        ]

        if condition_subset_df.empty:
            print("No transitions found for this specific condition.")
            continue

        count_matrix = pd.crosstab(condition_subset_df['From'], condition_subset_df['To'])
        prob_matrix = count_matrix.div(count_matrix.sum(axis=1), axis=0).fillna(0)

        print("\n--- Effective AOI Transition PROBABILITY Matrix ---")
        print(prob_matrix.to_string(float_format="%.2f"))

        descriptive_name = robot_name_map.get(robot, robot)

        plt.figure(figsize=(10, 8))
        sns.heatmap(prob_matrix, annot=True, fmt=".2f", cmap="YlGnBu", linewidths=.5,
vmin=0, vmax=1)
        plt.title(f"AOI Transition Probabilities for {descriptive_name} in '{difficulty}'
statements")

        plt.xlabel("To AOI")
        plt.ylabel("From AOI")
        plt.show()

# --- Main Execution Block ---
if __name__ == "__main__":
    run_full_transition_analysis(data_filepath=FULL_DATA_FILE,
                                 aoi_defs=AOI_DEFINITIONS)
```

First Python Script for Recurrence Quantification Anlaysis (Preparation)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pyrqa.time_series import TimeSeries
from pyrqa.settings import Settings
from pyrqa.analysis_type import Classic
from pyrqa.neighbourhood import FixedRadius
from pyrqa.metric import EuclideanMetric
from pyrqa.computation import RQAComputation
from pyrqa.image_generator import ImageGenerator
import traceback # For printing detailed error information

# --- USER: Define your parameters here ---

# 1. Name of the column containing the time series data you want to analyze for RQA
TIME_SERIES_COLUMN_FOR_RQA = 'Gaze point X (MCSnorm)'

# 2. RQA Parameters:
embedding_dim = 3
time_del = 10
threshold_radius_type = 'std_fraction'
threshold_value = 0.1

# Minimum number of data points in a trial required to perform RQA
MIN_DATA_POINTS_PER_TRIAL = 50

# --- Helper function to perform RQA ---
def calculate_rqa_for_series(series_data, emb_dim, t_delay, thresh_type, thresh_val):
    """Calculates RQA measures for a given time series."""
    if len(series_data) < MIN_DATA_POINTS_PER_TRIAL:
        print(f"    Skipping RQA: Not enough data points ({len(series_data)} <
{MIN_DATA_POINTS_PER_TRIAL})")
        return None, None

    time_series_obj = TimeSeries(series_data, embedding_dimension=emb_dim,
time_delay=t_delay)

    current_radius = 0.0
    if thresh_type == 'std_fraction':
        series_std = np.std(series_data)
        if series_std > 0:
            current_radius = thresh_val * series_std
        else:
            print(f"    Warning: Standard deviation is zero. Using a small fixed radius (0.01).")
```

```
            current_radius = 0.01
    elif thresh_type == 'fixed':
        current_radius = thresh_val
    else:
        print(f"    Warning: Unknown threshold_radius_type '{thresh_type}'. Defaulting to
'std_fraction'.")
        series_std = np.std(series_data)
        if series_std > 0:
            current_radius = thresh_val * series_std
        else:
            print(f"    Warning: Standard deviation is zero. Using a small fixed radius (0.01).")
            current_radius = 0.01

    if current_radius <= 0:
        print(f"    Warning: Calculated radius is non-positive ({current_radius}). Setting to a small
positive value (0.001).")
        current_radius = 0.001

    settings = Settings(time_series_obj,
                        analysis_type=Classic,
                        neighbourhood=FixedRadius(current_radius),
                        similarity_measure=EuclideanMetric,
                        theiler_corrector=1)
    try:
        computation = RQAComputation.create(settings, verbose=False)
        result = computation.run()

        rqa_measures = {
            'RR': result.recurrence_rate,
            'DET': result.determinism,
            'L_avg': result.average_diagonal_line,
            'L_max': result.longest_diagonal_line,
            'L_entr': result.entropy_diagonal_lines,
            'LAM': result.laminarity,
            'TT': result.trapping_time,
            'V_max': result.longest_vertical_line,
            'RP_threshold': current_radius
        }

        rp_matrix = None
        if hasattr(result, 'recurrence_matrix_reverse'):
            rp_matrix = result.recurrence_matrix_reverse
        else:
            print("    Warning: Recurrence matrix not found in result. Plotting will be skipped for this
trial.")
```

```
        return rqa_measures, rp_matrix

    except Exception as e:
        print(f"    An unrecoverable error occurred during RQA computation: {e}")
        return None, None


# --- Main analysis script ---
def main_analysis(csv_filepath, output_rqa_csv_file="rqa_results.csv",
example_plot_trial_id=1):
    """
    Main function to load data, preprocess, run RQA per trial, and save results.
    """
    print("Starting eye-tracking RQA analysis...")

    try:
        df = pd.read_csv(csv_filepath, decimal=',', na_values=['NA', ''])
        print(f"CSV data loaded successfully from: {csv_filepath}")
    except FileNotFoundError:
        print(f"Error: The file '{csv_filepath}' was not found.")
        return
    except Exception as e:
        print(f"Error loading CSV file '{csv_filepath}': {e}")
        return

    print("Performing initial data cleaning and preprocessing...")

    cols_to_convert_numeric = [
        'Gaze point X (MCSnorm)', 'Gaze point Y (MCSnorm)',
        'Pupil diameter left', 'Pupil diameter right',
        'Fixation point X (MCSnorm)', 'Fixation point Y (MCSnorm)'
    ]
    for col in cols_to_convert_numeric:
        if col in df.columns:
            df[col] = pd.to_numeric(df[col], errors='coerce')
        else:
            print(f"Warning: Expected numeric column '{col}' not found in CSV.")

    # UPDATED: Changed the required column name here
    required_columns = [TIME_SERIES_COLUMN_FOR_RQA, 'Eyetracker timestamp',
'robot_appearance_timeframe_number']
    for col in required_columns:
        if col not in df.columns:
            print(f"FATAL ERROR: A required column '{col}' is missing from the CSV. Cannot
proceed.")
            return
```

```
    df.dropna(subset=[TIME_SERIES_COLUMN_FOR_RQA, 'Eyetracker timestamp'],
inplace=True)
    print(f"  Rows after dropping essential NaNs (in '{TIME_SERIES_COLUMN_FOR_RQA}'
or 'Eyetracker timestamp'): {len(df)}")
    if df.empty:
        print("  DataFrame is empty after dropping essential NaNs. Cannot proceed.")
        return

    aoi_cols = ['is_cards', 'is_eyes', 'is_face', 'is_false_category',
                'is_robot', 'is_robot_name', 'is_true_category']
    for col in aoi_cols:
        if col in df.columns:
            if df[col].dtype == 'object':
                df[col] = df[col].str.lower().map({'true': True, 'false': False, '':
False}).fillna(False).astype(bool)
            else:
                df[col] = df[col].fillna(False).astype(bool)
        else:
            print(f"Warning: Expected AOI column '{col}' not found. It will be treated as False.")
            df[col] = False

    df['classification_category'] = (df.get('is_false_category', False) |
                                      df.get('is_true_category', False))
    print("  'classification_category' column created.")

    # UPDATED: Using the new timeframe column
    df['robot_appearance_timeframe_number'] =
pd.to_numeric(df['robot_appearance_timeframe_number'], errors='coerce')
    df.dropna(subset=['robot_appearance_timeframe_number'], inplace=True)
    if df.empty:
        print("  DataFrame is empty after dropping NA 'robot_appearance_timeframe_number'. No
trials to process.")
        return
    df['robot_appearance_timeframe_number'] =
df['robot_appearance_timeframe_number'].astype('Int64')
    print(f"  Rows after dropping NA trial numbers: {len(df)}")

    if 'ParticipantID' not in df.columns:
        print("Warning: 'ParticipantID' column not found. Creating a dummy 'Unknown'
ParticipantID.")
        df['ParticipantID'] = 'Unknown'
    else:
        df['ParticipantID'] = df['ParticipantID'].ffill().bfill()

    # UPDATED: Grouping for ffill now uses the new timeframe column
```

```
    grouping_cols_for_ffill = ['ParticipantID', 'robot_appearance_timeframe_number']
    cols_to_ffill = ['Robot', 'difficulty']
    for col_ffill in cols_to_ffill:
        if col_ffill in df.columns:
             df[col_ffill] = df.groupby(grouping_cols_for_ffill, group_keys=False)[col_ffill].ffill()
             df[col_ffill] = df.groupby(grouping_cols_for_ffill, group_keys=False)[col_ffill].bfill()
        else:
            print(f"Warning: Column '{col_ffill}' for IV not found. It will not be included in
results.")
            df[col_ffill] = 'N/A'

    print(f"  Using '{TIME_SERIES_COLUMN_FOR_RQA}' for RQA time series.")

    # --- 3. Perform RQA Trial-by-Trial ---
    all_rqa_results = []
    print("\nStarting RQA computation per trial...")

    # UPDATED: Main groupby now uses the new timeframe column
    grouped_trials = df.groupby(['ParticipantID', 'robot_appearance_timeframe_number'])

    for (participant_id, trial_num), trial_data in grouped_trials:
        print(f"\n  Processing Participant: {participant_id}, Trial: {trial_num}")

        robot_condition = trial_data['Robot'].iloc[0] if not trial_data['Robot'].empty else 'N/A'
        difficulty_level = trial_data['difficulty'].iloc[0] if not trial_data['difficulty'].empty else 'N/A'

        print(f"    Robot: {robot_condition}, Difficulty: {difficulty_level}")

        time_series_for_rqa = trial_data[TIME_SERIES_COLUMN_FOR_RQA].dropna().values

        rqa_output, rp_matrix = calculate_rqa_for_series(time_series_for_rqa,
                                                        embedding_dim,
                                                        time_del,
                                                        threshold_radius_type,
                                                        threshold_value)

        if rqa_output:
            print(f"    RQA successful for P{participant_id}, Trial {trial_num}.")
            trial_results = {
                'ParticipantID': participant_id,
                'Trial': trial_num,
                'Robot': robot_condition,
                'Difficulty': difficulty_level,
                'NumDataPoints': len(time_series_for_rqa),
                **rqa_output
            }
```

```
            all_rqa_results.append(trial_results)

            if trial_num == example_plot_trial_id and rp_matrix is not None:
                plot_filename =
f"recurrence_plot_participant_{participant_id}_trial_{trial_num}.png"
                try:
                    ImageGenerator.save_recurrence_plot(rp_matrix, plot_filename)
                    print(f"    Example recurrence plot saved as {plot_filename}")

                    img = plt.imread(plot_filename)
                    plt.figure(figsize=(6, 6))
                    plt.imshow(img, cmap='binary', origin='lower')
                    plt.title(f"RP: P{participant_id}, T{trial_num}
({TIME_SERIES_COLUMN_FOR_RQA})\nRobot: {robot_condition}, Diff:
{difficulty_level}")
                    plt.xlabel("Time Index")
                    plt.ylabel("Time Index")
                    plt.tight_layout()
                    plt.show()
                except Exception as e:
                    print(f"    Could not display/save example recurrence plot: {e}")
        else:
            print(f"    RQA failed or skipped for P{participant_id}, Trial {trial_num}.")

    # --- 4. Save Aggregated RQA Results ---
    if all_rqa_results:
        results_df = pd.DataFrame(all_rqa_results)
        results_df.to_csv(output_rqa_csv_file, index=False, decimal='.')
        print(f"\nAggregated RQA results saved to: {output_rqa_csv_file}")
        print("\n--- First 5 rows of RQA results ---")
        print(results_df.head())
        print("------------------------------------")
    else:
        print("\nNo RQA results were generated. Check data processing steps and trial lengths.")

    print("\nAnalysis complete.")
    print(f"Next steps: Analyze '{output_rqa_csv_file}' with your second script.")


# --- Run the analysis with your actual CSV file ---
if __name__ == "__main__":
    # Define the path to your data file and the name for your output file.
    actual_csv_filepath = "newest_combined_eyetracking_data.csv"
    output_filename = "robot_appearance_rqa_results_newest_data.csv"
    example_trial_to_plot = 1
```

```python
print(f"--- Attempting to run analysis with data from: {actual_csv_filepath} ---")

try:
    main_analysis(csv_filepath=actual_csv_filepath,
                  output_rqa_csv_file=output_filename,
                  example_plot_trial_id=example_trial_to_plot)
except FileNotFoundError:
    print(f"!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
    print(f"Error: The file '{actual_csv_filepath}' was not found.")
    print(f"Please ensure the file is in the same directory as the Python script,")
    print(f"or provide the full path to the file (e.g., '/path/to/your/{actual_csv_filepath}').")
    print(f"!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!")
except Exception as e:
    print(f"An unexpected error occurred during the analysis: {e}")
    traceback.print_exc()
```

Second Python Script for Recurrence Quantification Analysis (Exploration)

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import pingouin as pg

# --- SCRIPT INSTRUCTIONS ---
#
# 1.  Make sure your data file 'rqa_results_newest_data.csv' is in the same directory.
#
# 2.  Install necessary libraries if you haven't already:
#     pip install seaborn pingouin
#
# 3.  The script is set up to remove outliers > 2.5 SD from the mean. You can change
#     the SD_THRESHOLD variable if you wish.
#
# 4.  To analyze a different RQA measure, change the RQA_MEASURE_TO_ANALYZE
variable.
#
# 5.  Run this script from your terminal: python analyze_rqa_results.py

# --- USER CONFIGURATION ---
# The RQA measure you want to analyze from your CSV file
RQA_MEASURE_TO_ANALYZE = 'DET'  # Options: 'RR', 'DET', 'L_avg', 'LAM', 'TT', etc.

# The Standard Deviation threshold for outlier removal
SD_THRESHOLD = 2.5


def run_statistical_analysis(data_filepath, dv_measure, sd_thresh):
    """
    Loads RQA results, removes outliers using the SD method, and performs
    visualization and statistical analysis.
    """
    # --- Load the Data ---
    try:
        df = pd.read_csv(data_filepath)
        print(f"Successfully loaded RQA results from: {data_filepath}")
        if dv_measure not in df.columns:
            print(f"FATAL ERROR: The measure '{dv_measure}' is not a column in your data file.")
            print(f"Available columns are: {df.columns.tolist()}")
            return
    except FileNotFoundError:
        print(f"FATAL ERROR: The file '{data_filepath}' was not found.")
```

```
        return
    except Exception as e:
        print(f"An error occurred while loading the data: {e}")
        return

    print(f"\n--- Analysis started for RQA measure: {dv_measure} ---")


    # --- STEP 1: IDENTIFY AND REMOVE OUTLIERS ---
    print(f"\n[Step 1] Checking for outliers using the {sd_thresh} SD rule...")

    original_trial_count = len(df)

    def remove_outliers_by_sd(df, group_cols, value_col, threshold):
        """Identifies and removes outliers from a dataframe based on the SD rule."""
        def remove_group_outliers(group):
            mean = group[value_col].mean()
            std_dev = group[value_col].std()
            if pd.isna(std_dev) or std_dev == 0:
                return group
            lower_bound = mean - threshold * std_dev
            upper_bound = mean + threshold * std_dev
            return group[(group[value_col] >= lower_bound) & (group[value_col] <= upper_bound)]
        return df.groupby(group_cols, group_keys=False).apply(remove_group_outliers)

    df_cleaned = remove_outliers_by_sd(df,
                                       group_cols=['Robot', 'Difficulty', 'ParticipantID'],
                                       value_col=dv_measure,
                                       threshold=sd_thresh)

    final_trial_count = len(df_cleaned)
    outliers_removed_count = original_trial_count - final_trial_count

    if original_trial_count > 0:
        percentage_lost = (outliers_removed_count / original_trial_count) * 100
        print(f"  Original trial count: {original_trial_count}")
        print(f"  Removed {outliers_removed_count} outlier(s), which is {percentage_lost:.2f}% of
the data.")
        print(f"  Final trial count for analysis: {final_trial_count}")
    else:
        print("  No trials to process.")

    df = df_cleaned
    print("[Step 1] Outlier removal complete.")
```

```
# --- STEP 2: PREPARE DATA AND VISUALIZE ---
print("\n[Step 2] Preparing data and generating plots...")

# Define the mapping from old names to new, descriptive names
robot_name_map = {
    "Ryan condition": "Joint condition",
    "Ivan condition": "Disjoint condition",
    "Carl condition": "Control condition"
}
# Apply the mapping to the 'Robot' column
df['Robot'] = df['Robot'].map(robot_name_map)

# Define the desired order for the new names
robot_order = ["Joint condition", "Disjoint condition", "Control condition"]

# Check if all expected robot conditions are present in the data after mapping
actual_robots = df['Robot'].unique()
if all(robot in actual_robots for robot in robot_order):
    df['Robot'] = pd.Categorical(df['Robot'], categories=robot_order, ordered=True)
    print(f"  Condition names updated and custom plot order set: {robot_order}")
else:
    print(f"  Warning: Not all robots in 'robot_order' were found in the data after mapping.
Using default alphabetical order.")
    print(f"  Robots in data: {list(actual_robots)}")

sns.set(style="whitegrid", context="talk")

# Box plot for the main effect of 'Robot'
plt.figure(figsize=(12, 7))
sns.boxplot(x='Robot', y=dv_measure, data=df, palette="pastel")
sns.stripplot(x='Robot', y=dv_measure, data=df, color=".25", alpha=0.3)
plt.title(f'Effect of Robot Condition on {dv_measure} (Outliers Removed)')
plt.tight_layout()
plt.show()

# Box plot for the main effect of 'Difficulty'
plt.figure(figsize=(10, 7))
sns.boxplot(x='Difficulty', y=dv_measure, data=df, palette="pastel")
sns.stripplot(x='Difficulty', y=dv_measure, data=df, color=".25", alpha=0.3)
plt.title(f'Effect of Difficulty on {dv_measure} (Outliers Removed)')
plt.tight_layout()
plt.show()

# --- MODIFICATION START: Replaced interaction plot with a bar chart ---
# This bar chart shows the mean value for 'easy' and 'hard' conditions
# side-by-side for each robot condition.
```

```
    plt.figure(figsize=(12, 8))
    sns.barplot(x='Robot', y=dv_measure, hue='Difficulty', data=df,
                palette="colorblind", errorbar='se', capsize=.05)
    plt.title(f'Interaction of Robot and Difficulty on {dv_measure} (Outliers Removed)')
    plt.ylabel(f'Mean {dv_measure}')
    plt.legend(title='Difficulty')
    plt.tight_layout()
    plt.show()
    # --- MODIFICATION END ---

    print("[Step 2] Plots generated and displayed.")


    # --- STEP 3: PERFORM THE TWO-WAY REPEATED MEASURES ANOVA ---
    print(f"\n[Step 3] Performing Two-Way Repeated Measures ANOVA for '{dv_measure}'...")
    aov = pg.rm_anova(data=df, dv=dv_measure, within=['Robot', 'Difficulty'],
                      subject='ParticipantID', detailed=True)
    print("\n--- ANOVA Results ---")
    print(aov)


    # --- STEP 4: PERFORM POST-HOC TESTS (IF NECESSARY) ---
    is_robot_significant = aov.loc[aov['Source'] == 'Robot', 'p-unc'].iloc[0] < 0.05
    is_interaction_significant = aov.loc[aov['Source'] == 'Robot * Difficulty', 'p-unc'].iloc[0] <
0.05

    if is_robot_significant or is_interaction_significant:
        print(f"\n[Step 4] ANOVA showed significant effects. Performing post-hoc pairwise
tests...")
        posthocs = pg.pairwise_tests(data=df, dv=dv_measure, within=['Robot', 'Difficulty'],
                                     subject='ParticipantID', padjust='bonf')
        print("\n--- Post-Hoc Test Results ---")
        pd.set_option('display.max_rows', None)
        print(posthocs)
    else:
        print("\n[Step 4] No significant effects requiring post-hoc tests were found in the main
ANOVA.")


    # --- STEP 5: SIMPLE MAIN EFFECTS ANALYSIS ---

print("\n\n=============================================================
==========")
    print("[Step 5] Simple Main Effects: Testing Robot effect at each Difficulty Level")
```

```
print("=================================================================
=======\n")

    print("--- Analysis for 'hard' trials only ---")
    hard_df = df[df['Difficulty'] == 'hard'].copy()
    aov_hard = pg.rm_anova(data=hard_df, dv=dv_measure, within='Robot',
subject='ParticipantID', detailed=True)
    print("\n--- ANOVA for 'hard' trials only ---")
    print(aov_hard)
    posthocs_hard = pg.pairwise_tests(data=hard_df, dv=dv_measure, within='Robot',
subject='ParticipantID', padjust='bonf')
    print("\n--- Post-Hoc Tests for 'hard' trials only ---")
    print(posthocs_hard)

    print("\n\n--- Analysis for 'easy' trials only ---")
    easy_df = df[df['Difficulty'] == 'easy'].copy()
    aov_easy = pg.rm_anova(data=easy_df, dv=dv_measure, within='Robot',
subject='ParticipantID', detailed=True)
    print("\n--- ANOVA for 'easy' trials only ---")
    print(aov_easy)
    posthocs_easy = pg.pairwise_tests(data=easy_df, dv=dv_measure, within='Robot',
subject='ParticipantID', padjust='bonf')
    print("\n--- Post-Hoc Tests for 'easy' trials only ---")
    print(posthocs_easy)


    print(f"\n--- Analysis for '{dv_measure}' is complete. ---")


# --- Main Execution Block ---
if __name__ == "__main__":
    data_file = "robot_appearance_rqa_results_newest_data.csv"

    # Run the entire analysis workflow using the parameters from the top of the script
    run_statistical_analysis(data_filepath=data_file,
                             dv_measure=RQA_MEASURE_TO_ANALYZE,
                             sd_thresh=SD_THRESHOLD)
```