Measuring Visibility and Legibility of On-Screen Text Under Varying Font and Background Conditions: An Eyetracking Study

Khushboo Sharma

Measuring Visibility and Legibility of On-Screen Text Under Varying Font and Background Conditions: An Eyetracking Study

by

Khushboo Sharma

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Monday 22nd of September, 2025 at 13.45

Msc. Mechanical Engineering
BMD - Biorobotics Track

September 15, 2025

Student number:	4593529		
Thesis committee:	Yke Bauke Eisma,	TU Delft,	Main supervisor, Chairman
	Dimitra Dodou,	TU Delft,	Committee member
	Joost de Winter,	TU Delft,	External examiner

Faculty of Mechanical Engineering (ME)
Delft University of Technology

TU Delft Delft University of Technology

Measuring Visibility and Legibility of On-Screen Text Under Varying Font and Background Conditions: An Eyetracking Study

Abstract—This study investigates how font size, text-drawing style, and drop shadow affect the visibility, legibility, and comprehension of overlaid text. The purpose of this study is to better understand the role of these design factors and to offer practical guidelines for presenting information effectively on see-through displays. Twenty-five participants completed a visual search task, where they were asked to locate a target word presented under varying conditions on a complex background. A full factorial design was employed, incorporating four font sizes (0.10°, 0.15°, 0.20°, and 0.60°), six text-drawing styles (plain green text, plain white text, white text on blue billboards with 30%, 50%, 75% or 100% opacity), and the presence or absence of a drop shadow. Performance was evaluated across conditions in terms of noticeability (as a measure of visibility), processing time (as a measure of legibility), and word identification accuracy to determine significant differences. Applying a drop shadow improved legibility in plain text, while billboards lowered the upper performance threshold for font size. Although billboard conditions outperformed plain text conditions, varying billboard opacity had no significant effect on processing time, word identification, or noticeability. Overall, the findings suggest that font size, text styling, and background complexity interact to influence text visibility and legibility.

Index Terms — Legibility, Noteacibility, Processing time, Font size, Text-drawing style, Billboard, Complex background, See-through display

1 Introduction

Overlaid information on complex background has long been present in subtitles, video games, and other leisure contexts where the risks of reduced legibility are minimal. However, the increasing use of see-through displays, such as virtual reality (VR) glasses used in industrial workbenches (Di Donato et al., 2015) and teleoperated surgical procedures (Qian et al., 2017) and head-up displays (HUDs) in cars (Guo et al., 2020)(Ma et al., 2021), has introduced overlays into complex environments where safety is critical. In these contexts, overlaid information must integrate with the background. Poor contrast can degrade the legibility of the overlay, while excessive emphasis on the overlay can obscure important background details. Whereas in entertainment applications the problem can often be solved by simply adding a solid background or "billboard" behind the text, high-risk environments demand solutions that preserve the visibility of both the overlay and the underlying background to ensure safety (Horrey et al., 2006).

Legibility in human–machine interfaces (HMIs) refers to the ease with which users can read and understand information presented on a display while simultaneously engaging in other demanding tasks. As digital displays become increasingly integrated across domains, the volume and complexity of overlaid information continue to grow. In such contexts, legibility is not merely a matter of convenience but a critical factor for performance, efficiency, and safety. Poorly legible information can increase cognitive load, slow response times, and elevate the risk of errors, particularly in high-stakes environments where both the display content and the background scene carry essential information. While research on see-through displays is expanding, there remains a lack of studies focusing on the interaction of known legibility affecting variables, with most of the studies focusing solely on font size (Crundall et al., 2016) or solely on text-drawing style (Di Donato et al., 2015), and not taking complex backgrounds into consideration. Previous research on the long form of legibility has found that legibility is affected by various factors. Among these, typeface, font size, colour, contrast, and position are the most researched variables that play a key role in shaping visibility, legibility, and comprehension. Legibility has traditionally been measured in terms of reading speed, often quantified as words per minute. While it has been extensively studied in other fields (Rayner, 2009), much of the previous research has focused on simple, static backgrounds. However, complex dynamic backgrounds introduce challenges that have not previously been encountered. This study investigates how font size, text-drawing style, and dropshadow interact to affect users' ability to accurately and efficiently notice and process overlaid information in complex visual settings.

The paper is structured as follows: first, Section 2 discusses related work on legibility. Section 3 presents the method and describes the experimental design. Section 4 presents the results of the experiments conducted. Next, section 5 discusses the findings from the experiment. Section 6 gives the conclusions on the findings as well as the limitations of this study and possible directions for future research.

2 Related Works

Much of the research on legibility has traditionally focused on long-form reading on paper or on uniform, static backgrounds. However, reading in digital environments introduces additional variables that can affect readability, including background colour and complexity, stimulus position, luminance, and crowding. Although factors such as font size, font, contrast, line length, and case are known to affect legibility, their impact needs to be evaluated specifically for digital display environments. In this section, we review studies that examine how these factors influence legibility on digital displays.

2.1 Text Manipulation

Sawyer et al. (2017) studied the effect of text width and case by measuring the presentation duration threshold, while Beier & Oderkerk (2021) studied the letter stroke by measuring the letter recognition accuracy. Sawyer et al. (2017) found that the regular width had a significantly lower display threshold than the condensed case, and it was more pronounced in the condition of the smaller font size of 0.25° (3 mm) capital H-height. The uppercase typeface was found to be more legible as the display threshold was significantly lower than for the lowercase. Beier & Oderkerk (2021) varied the letter stroke by varying the font conditions that had either high, medium, or low stroke contrast. Results showed that the mean accuracy for the high letter stroke condition was significantly lower than the medium and low letter stroke conditions. The low letter stroke condition performed the best with the highest recognition accuracy. Studies exploring the effect of font (Reimer et al., 2014; Dobres et al., 2016) found that the font Frutiger performed well due to its open, unambiguous characters, as well as its compliance with the legibility characters described above. From the measured variables, Reimer et al. (2014) found a significantly lower response time, glance time and glance frequency for the font Frutiger than for Eurostile, while no significant difference was found for the error rate. Dobres et al. (2016) measured the response accuracy, response time and presentation duration threshold and found that the presentation duration threshold was significantly lower for the font Frutiger than for Eurostile, whereas no significant differences were found for response accuracy or response time. Regarding font sizes, previous studies such as Dobres et al. (2016)Sawyer et al. (2017) have studied the effect of 3 mm and 4 mm capital H-height, equivalent to 0.25° and 0.33°, respectively. Dobres et al. (2016) found a significantly lower display threshold for the 4 mm capital H-height condition compared to the 3 mm condition, while no significant difference was found for the response accuracy and the response time. Similarly, Sawyer et al. (2017) also found a significantly lower display threshold for the 4 mm condition compared to the 3 mm condition. While these studies studied the effect directly for digital display applications, they were still quite limited as they did not vary the background and simply studied plain text. Additionally, according to (Legge & Bigelow, 2011), there is a fluency range for the reading of text on displays, based on their x-height. The above studies do not take this range into account for their study, and only study 2 font sizes. Converting the capital H-height of Frutiger to x-height, using the reported ratio of 70.833% (F. Ltd., n.d.), gives 0.18° and 0.24° for the font sizes used in the studies above.

2.2 Background and Middle Layer Manipulation

Early research exploring the effects of background on legibility includes Gabbard et al. (2006), who displayed foreground text on see-through augmented reality (AR) glasses with outdoor textured wallpapers as backgrounds. They varied background textures and used static, including billboard style and plain green text, and active text-drawing styles. For the active style, the text color changed depending on the background. They reported interactions between background and text-drawing style, with active styles showing more pronounced effects. Performance depended strongly on the specific combination: for some backgrounds (red brick, granite, foliage), it remained fairly consistent across active styles, whereas for others (sky, pavement, sidewalk), it varied considerably depending on the text style. Overall, only billboard and green text were effective across all backgrounds. However, the backgrounds were not truly complex, and factors such as opacity and font size were not examined. Jankowski et al. (2010) extended this work by incorporating more complex backgrounds and examining plain and billboard-style text, but opacity and font size were still not varied. Their study employed a proofreading task, whereas applications in AR require short-form or glanceable reading where information must be quickly accessible. This study found no significant interactions. Both studies did not vary the position of the stimuli, and theerefore did not take the noticeability of the stimuli into account.

Gattullo et al. (2015) used industrial indoor backgrounds, including workbenches, and varied text colour and style (solid billboard and plain green text), finding no significant interaction between background and text style. For the stimuli, they used a short string of letters, more comparable to applications like AR. However, they did not vary text position, therefore not taking into account the noticeability of the stimuli. Kruijff et al. (2018) investigated complex indoor and outdoor data and measured the effect of backgrounds on label noticeability, looking at different label sizes for different coloured billboard text, including blue billboard. Opacity was not manipulated, and they found that background and region did not influence size selection for the optical see-through (OST) display. They also found no main effect of background on label noticeability. Interactions with text-drawing style were not analysed, and limited information was reported on the effect of label size.

More recent studies include Topliss et al. (2019) and Sawyer et al. (2020), who used complex backgrounds but did not account for position expectancy. As a result, these studies focused on processing time rather than noticeability and did not investigate interactions. Falk et al. (2021) tested white text on blue billboards with 50% opacity against a mixed-colour abstract background, but font size was not varied, and interactions were not examined. Hussain & Park (2023) employed an indoor background described as complex; however, large uniform areas made it appear almost uniform. They reported interactions between background and opacity.

Sawyer et al. (2020), Falk et al. (2021) and Hussain & Park (2023) all study the billboard opacity. Falk et al. (2021) only studied two levels of opacity (50% and 100%), while Hussain & Park (2023) tested five levels in increments of 25%. Despite the difference in the levels studied, both studies concluded that opacity level of 50% is the most ideal for a balanced visibility of both the foreground and background. While Falk et al. (2021) measured the search time for the letter N, Hussain & Park (2023) measured the error, task completion time and button visibility. Sawyer et al. (2017) measured the response accuracy as an indicator of legibility, where 80% was taken as the ideal legibility threshold. They measured scrim levels in increments of 15%, with a max of 60%. They found that 30% already reaches this threshold and simply recommend using this to avoid background visibility from being compromised. When using plain text, both Gabbard et al. (2006) and Debernardis et al. (2013) found that plain green text performs well in terms of the response time and error rates. Sawyer et al. (2020) also studied the effect of drop shadow on plain text and found that it achieved an accuracy level comparable to the ideal threshold reached with 30% scrim opacity.

Overall, while some studies employ complex backgrounds, few combine truly complex backgrounds with short-form reading tasks or examine critical factors such as font size, opacity, or position. While research suggests that drop shadows can improve legibility in complex environments, most existing studies have focused on billboards. Most research focuses on legibility, with very limited investigation of noticeability. Furthermore, none of these studies employed eye-tracking to directly measure visual attention or first fixation times. This highlights a gap in understanding how known factors affecting legibility and noticeability interact in realistic, complex environments.

3 METHODS

3.1 Participants

25 participants took part in the experiment (13 men, 9 women, and 3 non-binary). Participants' ages ranged from 17 to 54, with the mean age being 26.12 ($SD = 8.31$). Participants were screened for colour vision deficiencies using a six-item Ishihara test (Ishihara, 1972)(Bazilinskyy et al., 2020). The test revealed that two participants had minor red–green color deficiencies. They were not excluded from the analysis, as the deficiency was mild and the experiment does not have any conditions relying solely on red–green contrast. Moreover, including these participants was considered valuable for gaining additional insights. One participant reported being dyslexic. However, as typographical errors were accounted for using a typo-tolerant script in the later analysis, this participant was not excluded. One participant wore glasses during the study. The results showed no noticeable issues with tracking this participant, and therefore, this participant was not excluded from data analysis. All participants gave written informed consent to perform the experiment. The study was approved by the Human Research Ethics Committee of TU Delft, application number 5591.

3.2 Apparatus and Software

The experiment used a 24.5-inch monitor of the model BenqQ XL2540-B, a screen resolution of 1920 × 1080px, a display area of 541 x 301 mm, and a refresh rate of 120 Hz. The eye-to-screen distance was 93 cm. Eye movements were recorded using the SR Research EyeLink 1000 Plus in the head-stabilised position (S. R. Ltd., 2025), to maximise the sampling rate of the eye-tracker to 2000 Hz. The eye tracker was placed 53 cm from the chinrest and captured monocular eye movements of the right eye at a frequency of 2000 Hz. SR Research Experiment Builder was used for the experiment design. All experimental stimuli consisted of words systematically varied across the levels of the independent variables. Python was used for the generation of the stimuli and the background. Matlab was used for data processing and analysis.

3.3 Independent variables

The experiment included five independent variables: font size, text-drawing style, drop shadow, background pixel size, and stimulus position. Of these, font size (12, 18, 24, and 69 points - equivalant to 0.10°, 0.15°, 0.20°, and 0.60° respectively), text-drawing style (plain green text, plain white text, white text on blue billboards with 30%, 50%, 75% or 100% opacity), and drop shadow (present vs. absent) were the primary variables of interest. These factors were systematically controlled during the experiment. The colors for the stimuli were generated using the Pillow library's ImageColor module, with "lime" (RGB: 0, 255, 0) for the plain green text and "blue" (RGB: 0, 0, 255) for the billboard background. For conditions with drop shadows, the shadow was offset by 5 pixels horizontally and 5 pixels vertically relative to the text. To create randomised visual backgrounds, we generated pixelated noise images using a custom Python script (Pillow and NumPy libraries). Each background was initialised as an array of randomly sampled RGB values, with each channel uniformly drawn from the range of 0 to 255. The backgrounds were generated with pixel sizes ranging from 1 to 25, with each background corresponding to a single pixel size. This controlled the granularity of the noise, where smaller values produced fine-grained textures and larger values produced coarser, block-like patterns. This reflects diverse textures and object sizes, and therefore represents a large spectrum of background complexity. Expectancy increases the noticeability of events Wickens (2015). Therefore, we randomised the location of the stimuli. During the experiment, the position and the pre-generated backgrounds were randomised for the trials within Experiment Builder. By controlling the primary variables of interest, we were able to examine their effects while accounting for the variability introduced by the randomised factors.

3.4 Experimental task and procedure

All participants completed the experiment under the same conditions, in a dimly lit room. The illuminance of the room was within a range of 29.7-30.2 lux. Throughout the experiment, the background was uniformly grey (RGB: 125, 125, 125), except for the noisy background that contained the stimulus. The luminance of the noisy background was within a range of 41-58 cd/m2, and the luminance of the grey background was 56 cd/m2. The participants were provided with a short oral instruction before the start of the experiment. Once started, additional information and instructions were provided on the screen. The experiment employed a full-factorial within-

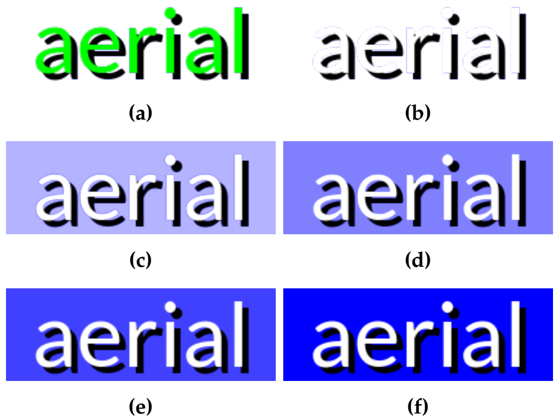


Figure 1: Overview of the six text-drawing styles used in the experiment with drop shadow. (a) Plain green text (b) Plain white text (c) White text on 30% opacity blue billboard (d) White text on 50% opacity blue billboard (e) White text on 75% opacity blue billboard (f) White text on 100% opacity blue billboard

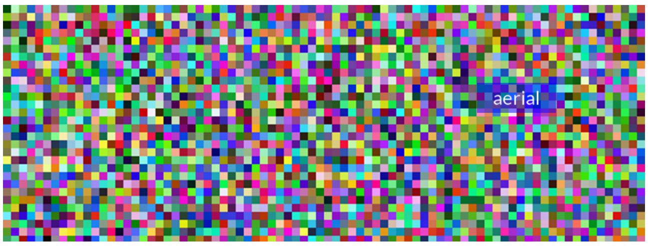


Figure 2: Stimuli overlaid on a complex, noisy background with pixel size 10 and 50% opacity billboard

subjects design. Participants were instructed to press the spacebar as soon as they located and read the target word presented on screen. On the next screen, they were required to type the word they had seen into a text field at their own pace. If they were unable to locate the word within a 30-second time limit, they were instructed to leave the text field blank. If they located the word but were unable to read it, they were asked to enter the letter "p" instead.

Each participant completed four blocks, corresponding to four different font sizes. At the beginning of each block, the eye tracker was recalibrated. Each block consisted of 36 trials, including 2 practice trials to familiarise participants with the task. The practice trials used a simple word and condition designed solely to illustrate the task. The practice trials used the font Arial at size 20. The stimulus word 'act' was presented on a billboard with 90% opacity against a 10 pixel background, while the position of the stimulus was randomized. This setup allowed participants to become familiar with the task without being exposed to the specific manipulations used in the main experiment. Each trial began with a fixation cross displayed for 2 seconds, which participants were instructed to focus on. At the beginning of each block, participants were allowed to take a break, during which they could remove their head from the headrest. Within each block, there were also 2 short breaks; however, during these breaks, participants were required to keep their heads in the headrest. The block order was pre-determined and counterbalanced across participants to control for order effects.

3.5 Dependent variables

The following dependent variables were measured during the study.

1) Accuracy (%): Represents the percentage of words read correctly. It serves as the first indicator of legibility, as low accuracy implies poor readability. Prior research (e.g., Legge & Bigelow (2011), Sawyer et al. (2020)) has often used 80% correct word recognition as a threshold for legibility. However, given our focus on safety-critical applications, we adopt a stricter criterion of 95% accuracy to ensure reliability.
2) Noticeability (ms): Defined as the time to the first meaningful fixation within the Area of Interest (AOI). First meaningful fixation was defined as the first fixation within the final cluster of fixations located inside the AOI prior to the spacebar press of the participant. Earlier AOI fixations were ignored if they were not part of this final sequence. This assumes that only the last sustained look at the AOI reflects the moment of stimulus recognition and subsequent processing. This measure indicates how quickly and easily the target stimuli can be located, providing insight into the influence of different experimental factors on visibility.
3) Processing time (ms): Processing time is calculated as the response time for the spacebar press minus noticeability. It reflects the time needed to interpret and respond to a stimulus once it has been detected, separating reading and cognitive processing from the initial search or detection phase.

3.6 Stimuli

Figure 3 shows the experimental setup. Stimulus location was randomised to avoid spatial expectancy. The stimuli were presented within an area ranging from 202 to 1717 pixels horizontally, and 62 to 1018 pixels vertically. These bounds were chosen to ensure that even the largest stimuli could be fully presented on screen without being clipped off, while still maintaining a margin from the edges. The stimuli and background were separately imported into the Experiment Builder software and programmed to be randomly combined. The location and condition of the stimuli were randomised using SR Research Experiment Builder. Previous research has shown that the valence of words affects the text processing and memory of words(Arfé et al., 2022). Therefore, neutral valence words were selected from Kousta et al. (2009). To maintain consistency in word length, only words containing six or seven letters were used. The font sizes that were used were based on the fluency angles for reading from Legge & Bigelow (2011). The fluent range of angular reading size in displays refers to the optimal angular size of text or content on a screen that ensures comfortable readability for users without causing strain or difficulty. This range depends on factors like viewing distance, font size, and display resolution. The fluency angle ranges from 0.2° to 2° when using test reading. For the RSVP method, where words are presented in isolation, the speed vs. printsize graph already starts declining after 1°. This makes sense, considering that for isolated words, there is only 1 fixation that samples the word, whereas the scrolling method used for larger text allows for more fixations to sample the words. While we are not using the RSVP method, we can argue that our method resembles this one more than the scrolling method, as the participants are instructed to press spacebar as soon as they located and read the word, and therefore do not allow for more fixations. However, from Figure 2 in Legge & Bigelow (2011) we can see that the peak for the RSVP method lies around 0.2°. This means that there is an upper critical threshold above which there is no longer an improvement in legibility. Using the Visual Angle Calculator (*Visual Angle Calculator - Calculator Academy*, 2023), we determined that at a viewing distance of 93 cm, 0.2° equals 3.2463 mm. The literature found that the better performing fonts had open and unambiguous characters with ample character spacing. Frutiger was specifically tested in multiple studies, however, since Frutiger is a commercial font, a free font with all the above properties was chosen, namely the font Lato by Google. According to Legge & Bigelow (2011), the point size that should be equivalent to fluent visual angles should be based on the x-height of the lowercase letter x. For our screen with a display area of 541 x 301 mm, and a resolution of 1920 × 1080 px, 1 mm = 3.61 px. Therefore, taking the x height of the font Lato 3.25 mm, we need a font size where the x-pixel size is equal to 12 px. This is equivalent to a font size of 24 pt. To evaluate whether the larger fonts indeed have a lower performance, we test the font size 69 pt equivalent to 0.60°. However, because the critical print size (CPS), below which the reading speed sharply declines, of Legge & Bigelow (2011) was based on plain text, this study was interested to examine if the CPS is affected by the presence of a billboard. Therefore, two smaller font sizes, 12 and 18 (corresponding to visual angles of 0.1° and 0.15°, respectively), were also included during this study.

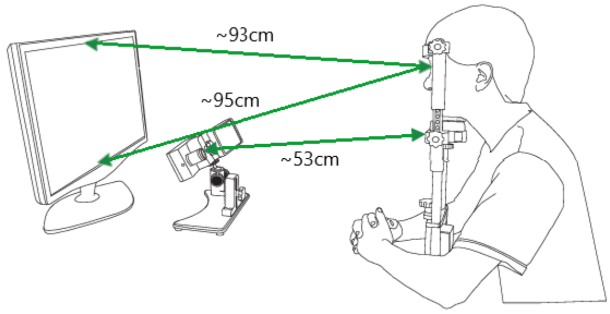


Figure 3: Setup of the experiment with the corresponding distances

3.7 Data Analysis

The study employed a within-subject, full factorial design. A custom fixation filter, based on Nyström & Holmqvist (2010) and used in previous studies (De Winter et al., 2023), was used to extract the fixations from the trials. A fixation, defined as a continuous gaze of at least 100 ms, with any overlapping blinks, was excluded from analysis. Normality of the data was assessed using the Shapiro–Wilk test. From the total of 3400 experimental trials across 25 participants, 695 of them were excluded for the noticeability and processing time analysis. Trials were excluded if no fixation occurred within the AOI, if the word was not successfully read when found, or if the trial duration was less than 100 ms. Because the inability to read or find a word is in itself an important metric, these trials were retained for the accuracy analysis. Therefore, the only exclusion criterion applied for the accuracy analysis was a trial duration of less than 100 ms. This resulted in 3396 trials for the accuracy analysis. To account for minor spelling errors in participants' typed responses, a typo-tolerance procedure was applied. A custom function compared each response to the correct target word using the Levenshtein edit distance. Responses were normalized (converted to lowercase and trimmed of whitespace), and cases where the participant gave up (empty response or entered "p") were marked as incorrect. A response was considered correct if the edit distance from the target was less than or equal to two character changes. This approach ensured that minor typographical errors did not artificially lower accuracy measures. Because of the unbalanced data that resulted from the trial exclusions, a Generalized Linear Mixed Model (GLMM) was used to assess the main and interacting effects. When significant effects were detected, post hoc pairwise comparisons were performed using the Wilcoxon signed-rank test. All post hoc results were adjusted for multiple comparisons using the Bonferroni-Holm correction. For the analysis of noticeability and processing time, the data was analyzed in two ways. For one analysis, the data was taken as a whole, while for the other, it was divided into plain text conditions and billboard conditions. To analyze noticeability and processing time, we fitted the GLMM with a gamma distribution and log link. For both models, font size, condition, and shadow, and their interactions were included as fixed effects, while the stimuli position X and Y served as covariates. Random intercepts were specified for participant, and random slopes were modeled within background. For the noticeability model, random slopes were included for condition, whereas for the processing time model, random slopes were included for font size, in order to improve the fit of each model. To further examine the influence of background, separate GLMM were fitted for both dependent variables. Font size and background were taken as fixed effects, while condition, shadow, and stimuli position were covariates. Random intercepts were specified for participants. All models were fitted using a gamma distribution with a log link function, and the significance of fixed effects was assessed via ANOVA. All analyses were conducted in MATLAB (version 2025a).

3.8 Hypothesis

Prior to conducting the study, we formulated the following hypotheses:

1) In the absence of a billboard, green text will be highly noticeable over complex backgrounds.
2) In the absence of a billboard, applying a drop shadow will significantly improve legibility, as measured by processing time.
3) Billboard opacity will have a significant effect on noticeability, while not significantly impacting processing time or accuracy percentages in word identification.
4) Noticeability is expected to increase with billboard opacity up to a threshold of 50%, beyond which no further improvement will occur.
5) The upper font size threshold beyond which there will be no further improvement in legibility, as measured by processing time, will be lower for the billboard condition compared to the plain text condition.

4 Results

This section presents the outcomes of the statistical analyses examining the effects of font size, text-drawing style, and drop shadow on noticeability, processing time, and error rate. The study employed a within-subject design with 25 participants.

4.1 Descriptive Statistics

Table 1 and Table 2 report mean values and standard deviations for noticeability and processing time across font sizes and conditions. Standard deviations indicate substantial variability across participants, particularly for noticeability.

4.2 Accuracy Percentage

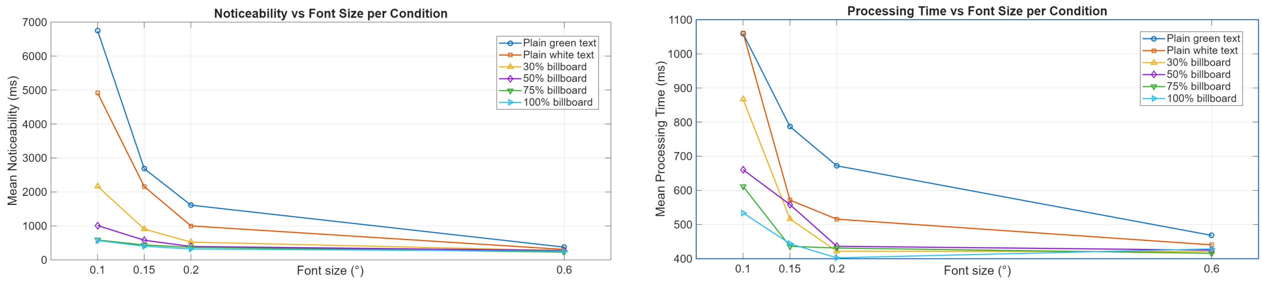
Figure 5 shows us the accuracy analysis of all the conditions across the font sizes. It is clear that the accuracy percentage is overall quite high. For font size 12, the plain green and white text, and 30% opacity billboard conditions had an accuracy under 95%. Font size 18 with condition green NS also performed below the accuracy threshold of 95%. Except for font size 12 and the green NS condition for fontsize 18, the text-drawing style had no effect on the accuracy of the word identification. For the analysis where the data was divided into plain text and billboard conditions, font size 12 was excluded because of its low accuracy.

Table 1: Mean and SD of processing times and noticeability per font size.

Font Size	Noticeability Mean	ProcessingTime Mean	Noticeability SD	ProcessingTime SD
f12	2480.8	786.27	4310.2	1003.6
f18	1195.5	553.02	2366.5	580.89
f24	708.85	480.73	1388.8	449.55
f69	286.32	433.59	236.4	399.86

Table 2: Mean and SD of processing times and noticeability per text-drawing style.

Condition	Noticeability Mean	ProcessingTime Mean	Noticeability SD	ProcessingTime SD
Green text	2463	712.22	4459.9	866.89
White text	1913.5	624.13	3511	911.52
30% billboard	926.2	546.29	1663.9	629.56
50% billboard	554.8	516.57	790.77	516.29
75% billboard	394.9	468.5	454.98	351.87
100% billboard	380.95	450.1	660.7	309.94



(a)

(b)

Figure 4: Interaction effects of Font size and Condition shown for (a) Noticeability and (b) Processing Time.

4.3 Noticeability

The ANOVA results of the GLMM gamma-log model with *font size* (f12, f18, f24, f69), *text-drawing style* (green, white, 30% billboard , 50% billboard, 75% billboard, 100% billboard), and *drop shadow* (present, absent) as independent variables, are depicted in Table 3. Table 4 shows the post-hoc results. Table 5 and 6 show the ANOVA results for the separated plain text and billboard analysis, where font size 12 is excluded.

- The full ANOVA results (Table 3) indicate significant main effects for font size ($p < 0.001$, $\eta^2 = 0.100$), condition ($p < 0.001$, $\eta^2 = 0.088$) and their interaction ($p < 0.001$, $\eta^2 = 0.079$). Figure 4a additionally shows the relationship between font size and condition for noticeability. While other interactions were also significant, their effect sizes were low. Post-hoc comparisons further revealed that font size 12 performed significantly worse than the other font sizes, whereas font size 69 resulted in significantly better performance for noticeability. In addition, both plain text conditions performed significantly worse than all billboard conditions, while no significant difference in performance was found between the billboard conditions.
- Plain white text ($M = 1913.5$, $SD = 3511$) performed better than plain green text ($M = 2463$, $SD = 4459.9$). However, the result was not statistically significant.
- The main effect of font size on noticeability for plain text was significant, $p < 0.001$, with the largest observed effect size, $\eta^2 = 0.186$. Drop shadow also had a significant effect, $p < 0.001$, $\eta^2 = 0.023$. The other variables were also significant, however, the effect sizes were low.
- For the billboard conditions, font size remained the largest effect for noticeability, $\eta^2 = 0.056$, $p < 0.001$, while text-drawing style also had a moderate effect, $p < 0.001$, $\eta^2 = 0.038$, and the interaction of font size and style was also significant, $p < 0.001$, $\eta^2 = 0.020$.

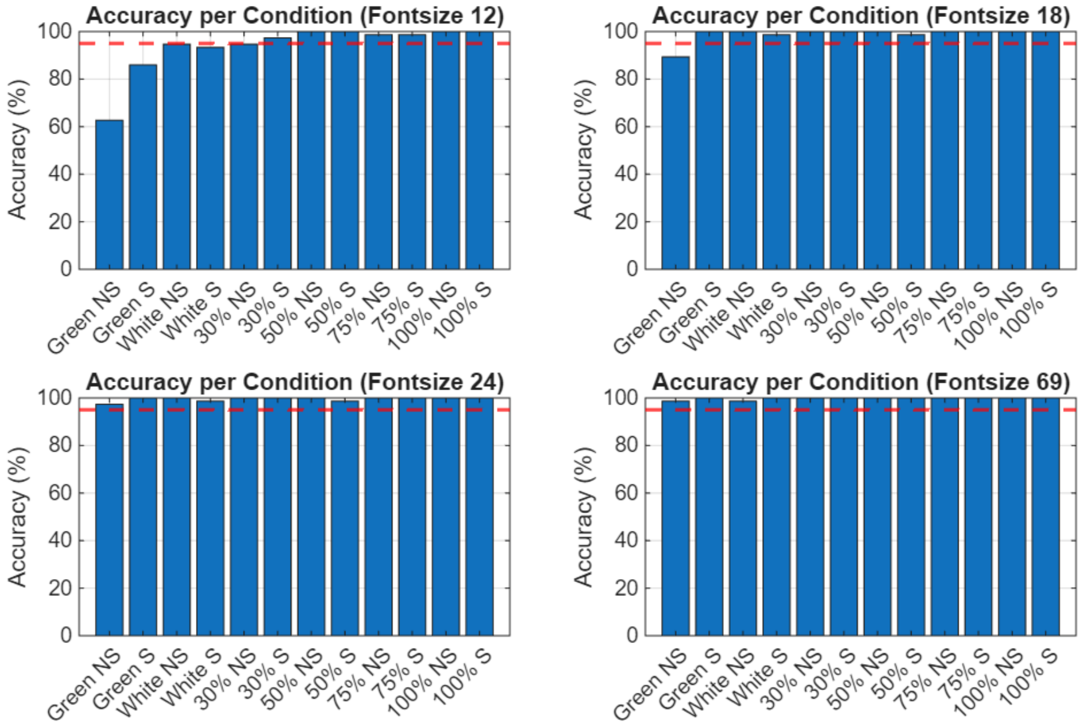


Figure 5: Accuracy (%) per font size and condition. The red dashed line indicates the 95% threshold.

Table 3: ANOVA results for Noticeability and Processing Time for $FontSize \times Condition \times Shadow$. Partial η^2 values reported. Significant p-values are marked in bold and p <0.001 are marked with *.

Term	DF1	DF2	Noticeability			Processing Time		
			F	p	η_p^2	F	p	η_p^2
Font Size	3	2655	97.885	**0.001***	0.100	24.13	**0.001***	0.027
Condition	5	2655	50.963	**0.001***	0.088	17.868	**0.001***	0.033
Shadow	1	2655	0.613	0.434	0.0002	1.387	0.239	0.001
Font Size:Condition	15	2655	15.177	**0.001***	0.079	3.731	**0.001***	0.021
Font Size:Shadow	3	2655	3.500	**0.015**	0.004	2.832	**0.037**	0.003
Condition:Shadow	5	2655	3.554	**0.003**	0.007	2.016	0.073	0.004
Font Size:Condition:Shadow	15	2655	2.769	**0.001***	0.015	1.621	0.061	0.009

4.4 Processing Time

A similar analysis was conducted for processing time.

- The full ANOVA results (Table 3) indicate significant main effects of font size ($p < 0.001$, $\eta^2 = 0.027$), condition ($p < 0.001$, $\eta^2 = 0.033$), and the interaction between font size and condition ($p < 0.001$, $\eta^2 = 0.021$). Figure 4b, additionally shows the relationship between font size and condition for processing time. Post-hoc comparisons revealed that for the font size, there was only a significant difference between the pair f12 and f24, and f12 and f69. For the condition factor, all plain green text conditions differed significantly from the billboard conditions, except for the 30% opacity level. The plain white text condition only differed significantly from the full opacity billboard.
- For the plain text analysis, a significant main effect was found for font size ($p < 0.001$, $\eta^2 = 0.054$), condition ($p = 0.004$, $\eta^2 = 0.012$) as well as drop shadow ($p = 0.002$, $\eta^2 = 0.013$). Post hoc comparisons indicated a significant difference between f18 and f24 ($p = 0.002$), and f18 and f24 ($p = 0.005$).
- For the billboard analysis, only a main effect of font size, $p < 0.001$, $\eta^2 = 0.023$, was found. However, post hoc Holm-Bonferroni corrected values, revealed no significant differences in pairwise comparisons. The billboard opacity had no significant effect on the processing time.

Table 4: Post-hoc Wilcoxon signed-rank tests for Noticeability (N) and Processing Time (P). Only significant Holm-Bonferroni corrected p-values are reported. P<0.001 are marked with *.

Comparison	Corrected p (N)	Corrected p (P)
f12 vs f18	0.034	-
f12 vs f24	0.001*	0.001*
f12 vs f69	0.001*	0.001*
f18 vs f69	0.001	-
f24 vs f69	0.003	-
Green vs 30% billboard	0.003	-
Green vs 50% billboard	0.001*	0.018
Green vs 75% billboard	0.001*	0.002
Green vs 100% billboard	0.001*	0.002
White vs 30% billboard	0.009	-
White vs 50% billboard	0.003	-
White vs 75% billboard	0.001*	-
White vs 100% billboard	0.001*	0.047

Table 5: ANOVA results for plain text condition Noticeability (N) and Processing Time (P) for $FontSize \times Condition \times Shadow$, with f18, f24, f69. Partial η^2 values are reported. Significant p-values are marked in bold, and p <0.001 are marked with *.

Term	DF1	F (N)	DF2 (N)	p (N)	η_p^2 (N)	F (P)	DF2 (P)	p (P)	η_p^2 (P)
Font Size	2	80.124	699	**0.001***	0.186	19.853	698	**0.001***	0.054
Condition	1	10.608	699	**0.001**	0.015	8.2717	698	**0.004**	0.012
Shadow	1	16.166	699	**0.001***	0.023	9.4243	698	**0.002**	0.013
Font Size:Condition	2	3.7589	699	**0.024**	0.011	0.90956	698	0.403	0.003
Font Size:Shadow	2	3.4343	699	**0.033**	0.010	2.5504	698	0.079	0.007
Condition:Shadow	1	5.8746	699	**0.016**	0.008	0.75239	698	0.386	0.001
Font Size:Condition:Shadow	2	3.9168	699	**0.020**	0.011	0.87085	698	0.419	0.002

Table 6: ANOVA results for billboard Noticeability (N) and Processing Time (P) for $FontSize \times Condition \times Shadow$, with f18, f24, f69. Partial η^2 values are reported. Significant p-values are marked in bold and p <0.001 are marked with *.

Term	DF1	F (N)	DF2 (N)	p (N)	η_p^2 (N)	F (P)	DF2 (P)	p (P)	η_p^2 (P)
Font Size	2	41.056	1376	**0.001***	0.056	16.404	1373	**0.001***	0.023
Condition	3	18.33	1376	**0.001***	0.038	2.4982	1373	0.058	0.005
Shadow	1	4.0563	1376	**0.044**	0.003	1.0581	1373	0.304	0.001
Font Size:Condition	6	4.8065	1376	**0.001***	0.020	1.1886	1373	0.310	0.005
Font Size:Shadow	2	2.8127	1376	0.060	0.004	0.97196	1373	0.379	0.001
Condition:Shadow	3	1.4794	1376	0.218	0.003	1.0571	1373	0.366	0.002
Font Size:Condition:Shadow	6	1.3136	1376	0.248	0.006	1.0795	1373	0.373	0.005

4.5 Background

Figure 6 and Figure 7 show us the relationship between background and font size. From Table 7 we can see that background has a significant effect on both the noticeability and processing time, $p < 0.001$ and $\eta_p^2 = 0.045$ and $\eta_p^2 = 0.047$ respectively, while font size only has a significant effect on processing time. The interaction also has the largest effect for both noticeability and processing time, with $p < 0.001$, and $\eta_p^2 = 0.057$ and $\eta_p^2 = 0.052$, respectively.

5 Discussion

This study examined the effects of font size, text-drawing style, and billboard opacity on noticeability and legibility against complex backgrounds. Compared to Sawyer et al. (2020), accuracy percentages for the small-font conditions in our study were relatively high. This is likely due to our self-paced experimental design.

5.1 Text-drawing style

While previous literature, such as Gabbard et al. (2006) and Debernardis et al. (2013), suggests that green is the most effective colour when no billboard is used, this paper did not find a similar pattern. On the contrary, the mean processing time and noticeability for the plain green text condition were higher than the plain white text, though the results were not significant. This find-

Table 7: ANOVA results for Noticeability (N) and ProcessingTime (P) for $Background \times Fontsize$. Significant p-values are marked in bold and p <0.001 are marked with *.

Term	DF1	DF2	Noticeability			Processing Time		
			F	p	η_p^2	F	p	η_p^2
Font Size	3	2597	0.423	0.737	0.0005	15.724	**0.001***	0.018
Background	24	2597	5.124	**0.001***	0.045	5.351	**0.001***	0.047
Font Size:Background	72	2597	2.176	**0.001***	0.057	1.965	**0.001***	0.052

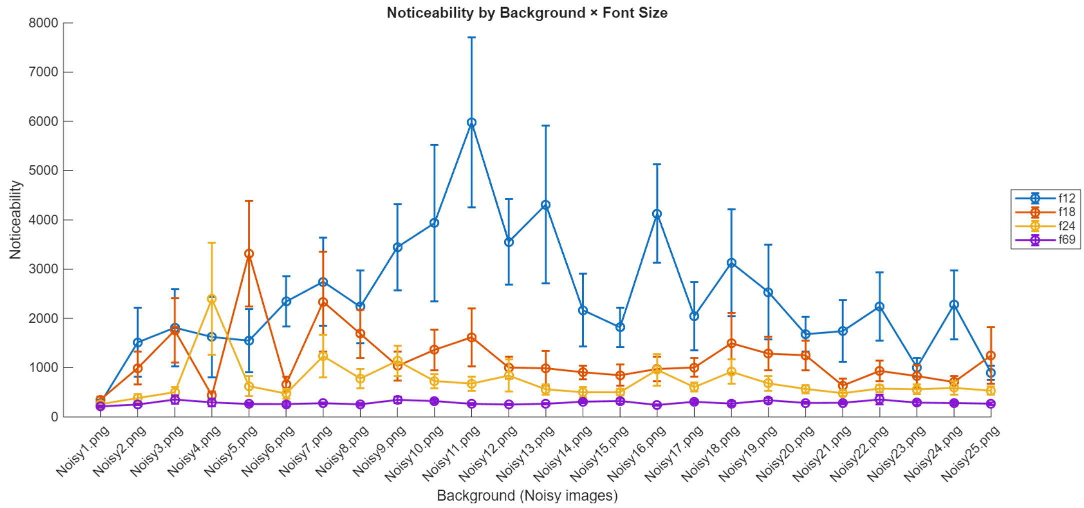


Figure 6: Interaction effects of Background x Font size for noticeability.

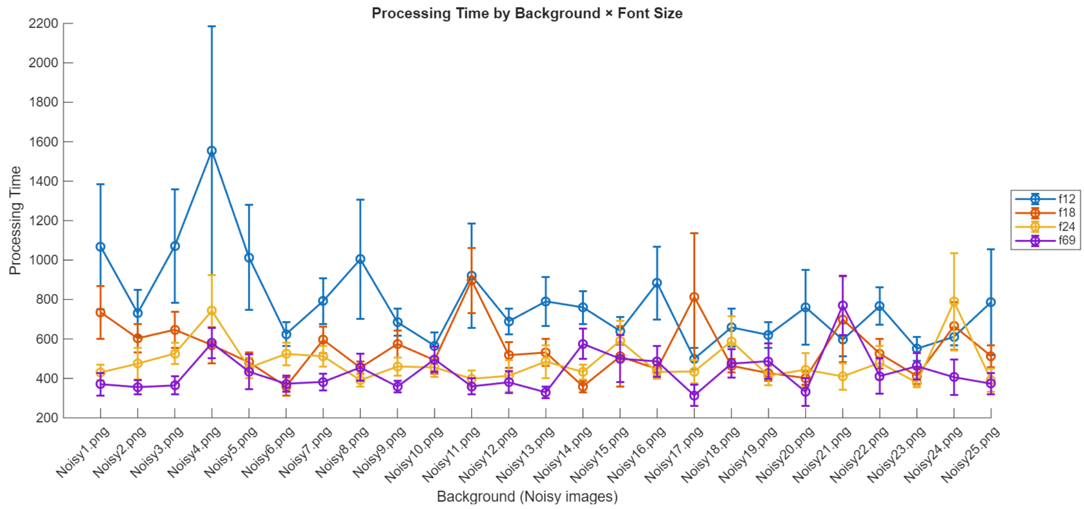


Figure 7: Interaction effects of Background x Font size for processing time.

ing is particularly interesting given that green text remains the standard for HUDs in aircraft.

This difference in finding can be explained by differences in experimental conditions. Prior studies, including Gabbard et al. (2006), typically used textured or patterned backgrounds that were not highly complex. In contrast, the current experiment employed a random noise background with varying pixel sizes, providing a truly complex visual environment.

Regarding billboard opacity, no significant differences were observed in noticeability or processing time, contrary to our hypothesis H4. This suggests that reducing billboard opacity does not negatively affect the legibility or visibility of the stimuli. Consequently, unlike previous findings from Falk et al. (2021) and Hussain & Park (2023), our results indicate that billboard opacity can be safely reduced to 30% without affecting performance. This is particularly relevant in applications where the stimulus location is already known, such as on a dashboard, where noticeability becomes less critical. For these applications, it may even be possible to reduce opacity further.

5.2 Font size

Previous research on font size has largely focused on long-form reading contexts. While some studies, such as Dobres et al. (2016) and Sawyer et al. (2017), have examined font size in glanceable legibility tasks, they tested only two sizes—3 mm and 4 mm capital letter height. The measures reported in those studies are conceptually equivalent to our processing time metric. They observed a significant improvement in legibility for plain text from 3 mm to 4 mm, equivalent to 0.18° and 0.24°, respectively. Figure 4b shows the interaction plot for processing time between font size and condition. We can see in Figure 4b that the decline starts to slow from font size 0.2°. The statistical analysis found a significant difference in processing time between f18 and f24 (0.15° and 0.20°), for the plain text condition, but did not reveal a significant difference between f24 and f69 conditions. This finding agrees with Legge & Bigelow (2011) where the peak performance for reading speed lies around 0.20° for plain text. In the same Figure 4b, we see that the slope of decline for the processing time is much lower after 0.15° for the billboard conditions. The statistical analysis confirmed that there was no significant difference for the Holm-Bonferroni pairwise comparison between the font sizes f18, f24 and f69. This indicates that performance does not improve further once font size exceeds 0.15° in the billboard condition. This finding supports H5, and shows that there is indeed a smaller upper font size threshold for the billboard condition compared to the plain text.

5.3 Dropshadow

We found that drop shadow significantly affected the processing time for stimuli when using plain text, thereby supporting our hypothesis. From the factors examined, drop shadow produced the second-largest effect size, suggesting that, when adjusting font size is not an option for plain text, adding a drop shadow can provide a subtle improvement in legibility. Interestingly, our results also revealed a main effect of drop shadow on noticeability for plain text, with a fairly large effect size. One explanation for this could be that applying a drop shadow effectively adds an edge-like boundary around text, which could have boosted noticeability by improving separation and emphasizing contours.

5.4 Background

From the analysis, an interaction effect for background and font size was found for both the noticeability and the processing time. However, because of the large number of backgrounds, the post-hoc pairwise comparison did not result in a reliable outcome. From Figure 6, we can see that for font sizes 18 and 24, the noticeability time lies higher for smaller pixel sizes, whereas for font size 12, the noticeability gradually increases and reaches a peak at pixel size 11, after which it declines again. This effect is caused by a size-based interference. When the font size approaches the same scale as the background's pixel noise, a crowding effect occurs, reducing noticeability Whitney & Levi (2011). For the processing time, Figure 7 reveals that font size 12 is less legible when the background has small pixel sizes. Both Figure 6 and Figure 7 reveal that the interaction between font size and background stabilises as the font size increases.

6 Conclusion

This paper evaluated the effects of font size and text-drawing style on noticeability and processing time. Hypothesis H1, predicting higher noticeability for green text over complex backgrounds without a billboard, was not supported. Hypothesis H4, predicting an increase in noticeability until a threshold of billboard opacity 50%, was also not supported. Hypotheses H2 and H5 were confirmed: applying a drop shadow improved legibility in plain text (H2) and applying a billboard decreased the upper performance threshold for font size (H5). Hypothesis H3 was partially confirmed. As predicted, billboard opacity did not significantly affect processing time or word identification accuracy. However, contrary to our expectations, it also did not affect noticeability. The findings in this study suggest that while applying billboards and larger fonts can enhance text visibility, the interaction between background complexity and font size also plays a significant role.

This study is subject to several limitations. With only 19 words available, repetition was necessary and likely introduced a learning effect. Additionally, in scenarios where word positions are fixed, differences in noticeability may be reduced, as expectancy has a strong influence on visual scanning. Another limitation is that the study did not account for visual clutter. Since real-world applications frequently involve visually complex environments, clutter is expected to have a considerable impact. Future research should therefore extend these findings by examining noticeability and processing time under more crowded visual conditions. Furthermore, this study was conducted in a dimly lit environment. Future work could build on these findings by repeating the experiment under brighter lighting conditions to study how increased luminance affects noticeability and processing time. Finally, while this study examined the effect of opaque billboard conditions on legibility and noticeability in order to preserve background visibility, the background visibility itself was not directly studied. Therefore, future work should explicitly evaluate the impact of these conditions on background visibility.

References

Arfé, B., Delatorre, P., & Mason, L. (2022, 10). Effects of negative emotional valence on readers' text processing and memory for text: an eye-tracking study. *Reading and Writing*, *36*(7), 1743–1768. Retrieved from https://doi.org/10.1007/s11145-022-10362-7 doi: 10.1007/s11145-022-10362-7

Bazilinskyy, P., Dodou, D., & De Winter, J. (2020, 10). External Human-Machine Interfaces: Which of 729 Colors Is Best for Signaling 'Please (Do not) Cross'? *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 3721–3728. Retrieved from https://doi.org/10.1109/smc42975.2020.9282998 doi: 10.1109/smc42975.2020.9282998

Beier, S., & Oderkerk, C. A. (2021, 6). High letter stroke contrast impairs letter recognition of bold fonts. *Applied Ergonomics*, *97*, 103499. Retrieved from https://doi.org/10.1016/j.apergo.2021.103499 doi: 10.1016/j.apergo.2021.103499

Crundall, E., Large, D. R., & Burnett, G. (2016, 5). A driving simulator study to explore the effects of text size on the visual demand of in-vehicle displays. *Displays*, *43*, 23–29. Retrieved from https://doi.org/10.1016/j.displa.2016.05.003 doi: 10.1016/j.displa.2016.05.003

Debernardis, S., Fiorentino, M., Gattullo, M., Monno, G., & Uva, A. E. (2013, 5). Text Readability in Head-Worn Displays: Color and Style Optimization in Video versus Optical See-Through Devices. *IEEE Transactions on Visualization and Computer Graphics*, *20*(1), 125–139. Retrieved from https://doi.org/10.1109/tvcg.2013.86 doi: 10.1109/tvcg.2013.86

De Winter, J. C. F., Dodou, D., & Eisma, Y. B. (2023, 6). Responses to Raven matrices: Governed by visual complexity and centrality. *Perception*, *52*(9), 645–661. Retrieved from https://doi.org/10.1177/03010066231178149 doi: 10.1177/03010066231178149

Di Donato, M., Fiorentino, M., Uva, A. E., Gattullo, M., & Monno, G. (2015, 3). Text legibility for projected Augmented Reality on industrial workbenches. *Computers in Industry*, *70*, 70–78. Retrieved from https://doi.org/10.1016/j.compind.2015.02.008 doi: 10.1016/j.compind.2015.02.008

Dobres, J., Chahine, N., Reimer, B., Gould, D., Mehler, B., & Coughlin, J. F. (2016, 1). Utilising psychophysical techniques to investigate the effects of age, typeface design, size and display polarity on glance legibility. *Ergonomics*, *59*(10), 1377–1391. Retrieved from https://doi.org/10.1080/00140139.2015.1137637 doi: 10.1080/00140139.2015.1137637

Falk, J., Eksvard, S., Schenkman, B., Andren, B., & Brunnstrom, K. (2021, 6). Legibility and readability in Augmented Reality. *International Conference on Quality of Multimedia Experience (QoMEX)*, 231–236. Retrieved from https://doi.org/10.1109/qomex51781.2021.9465455 doi: 10.1109/qomex51781.2021.9465455

Gabbard, J. L., Swan, J. E., & Hix, D. (2006, 2). The effects of text drawing styles, background textures, and natural lighting on text legibility in outdoor augmented reality. *PRESENCE Virtual and Augmented Reality*, *15*(1), 16–32. Retrieved from https://doi.org/10.1162/pres.2006.15.1.16 doi: 10.1162/pres.2006.15.1.16

Gattullo, M., Uva, A. E., Fiorentino, M., & Gabbard, J. L. (2015, 3). Legibility in industrial AR: text style, color coding, and illuminance. *IEEE Computer Graphics and Applications*, *35*(2), 52–61. Retrieved from https://doi.org/10.1109/mcg.2015.36 doi: 10.1109/mcg.2015.36

Guo, W., Yan, D., Liu, T., & Zhang, Z. (2020, 12). Technical challenge and solution for vehicle-mounted AR-HUD mass commercial application. *International Conference on Optoelectronic and Microelectronic Technology and Application*, 187. Retrieved from https://doi.org/10.1117/12.2586525 doi: 10.1117/12.2586525

Horrey, W. J., Wickens, C. D., & Consalus, K. P. (2006, 1). Modeling drivers' visual attention allocation while interacting with in-vehicle technologies. *Journal of Experimental Psychology Applied*, *12*(2), 67–78. Retrieved from https://doi.org/10.1037/1076-898x.12.2.67 doi: 10.1037/1076-898x.12.2.67

Hussain, M., & Park, J. (2023, 5). Effect of transparency levels and Real-World backgrounds on the user interface in augmented reality environments. *International Journal of Human-Computer Interaction*, *40*(16), 4265–4274. Retrieved from https://doi.org/10.1080/10447318.2023.2212218 doi: 10.1080/10447318.2023.2212218

Ishihara, S. (1972). Tests for colour-blindness. *Tokyo: Kanehara Shuppan Co., Ltd.* Retrieved from https://web.archive.org/web/20201208160704/http://www.dfisica.ubi.pt/~hgil/p.v.2/Ishihara/Ishihara.24.Plate.TEST.Book.pdf

Jankowski, J., Samp, K., Irzynska, I., Jozwowicz, M., & Decker, S. (2010, 4). Integrating Text with Video and 3D Graphics. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. Retrieved from https://doi.org/10.1145/1753326.1753524 doi: 10.1145/1753326.1753524

Kousta, S.-T., Vinson, D. P., & Vigliocco, G. (2009, 7). Emotion words, regardless of polarity, have a processing advantage over neutral words. *Cognition*, *112*(3), 473–481. Retrieved from https://doi.org/10.1016/j.cognition.2009.06.007 doi: 10.1016/j.cognition.2009.06.007

Kruijff, E., Orlosky, J., Kishishita, N., Trepkowski, C., & Kiyokawa, K. (2018, 7). The influence of label design on search performance and noticeability in wide field of view augmented reality displays. *IEEE Transactions on Visualization and Computer Graphics*, *25*(9), 2821–2837. Retrieved from https://doi.org/10.1109/tvcg.2018.2854737 doi: 10.1109/tvcg.2018.2854737

Legge, G. E., & Bigelow, C. A. (2011, 8). Does print size matter for reading? A review of findings from vision science and typography. *Journal of Vision*, *11*(5), 8. Retrieved from https://doi.org/10.1167/11.5.8 doi: 10.1167/11.5.8

Ltd., F. (n.d.). *Setting your letter Heights - FontLab 8.* Retrieved from https://help.fontlab.com/fontlab/8/tutorials/calfonts/3.%20Fitting%20and%20Spacing/03a%20Setting%20Your%20Letter%20Heights/#:~:text=The%20More%20Modern%20Take,is%2070.833%25%20of%20the%20uppercase.&text=The%20area%20is%2050.17%25.,changed%20to%20one%20of%20area.

Ltd., S. R. (2025, 8). *EyeLink 1000 Plus - Eye Tracker - Fast, accurate, reliable eye tracking.* Retrieved from https://www.sr-research.com/eyelink-1000-plus/

Ma, X., Jia, M., Hong, Z., Kwok, A. P. K., & Yan, M. (2021, 1). Does Augmented-Reality Head-Up Display help? a preliminary study on driving performance through a VR-Simulated Eye Movement analysis. *IEEE Access*, *9*, 129951–129964. Retrieved from https://doi.org/10.1109/access.2021.3112240 doi: 10.1109/access.2021.3112240

Nyström, M., & Holmqvist, K. (2010, 2). An adaptive algorithm for fixation, saccade, and glissade detection in eyetracking data. *Behavior Research Methods*, *42*(1), 188–204. Retrieved from https://doi.org/10.3758/brm.42.1.188 doi: 10.3758/brm.42.1.188

Qian, L., Barthel, A., Johnson, A., Osgood, G., Kazanzides, P., Navab, N., & Fuerst, B. (2017, 3). Comparison of optical see-through head-mounted displays for surgical interventions with object-anchored 2D-display. *International Journal of Computer Assisted Radiology and Surgery*, *12*(6), 901–910. Retrieved from https://doi.org/10.1007/s11548-017-1564-y doi: 10.1007/s11548-017-1564-y

Rayner, K. (2009, 5). The 35th Sir Frederick Bartlett Lecture: Eye movements and attention in reading, scene perception, and visual search. *Quarterly Journal of Experimental Psychology*, *62*(8), 1457–1506. Retrieved from https://doi.org/10.1080/17470210902816461 doi: 10.1080/17470210902816461

Reimer, B., Mehler, B., Dobres, J., Coughlin, J. F., Matteson, S., Gould, D., ... Levantovsky, V. (2014, 7). Assessing the impact of typeface design in a text-rich automotive user interface. *Ergonomics*, *57*(11), 1643–1658. Retrieved from https://doi.org/10.1080/00140139.2014.940000 doi: 10.1080/00140139.2014.940000

Sawyer, B. D., Dobres, J., Chahine, N., & Reimer, B. (2017, 9). The Cost of Cool: Typographic style legibility in reading at a Glance. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, *61*(1), 833–837. Retrieved from https://doi.org/10.1177/1541931213601698 doi: 10.1177/1541931213601698

Sawyer, B. D., Wolfe, B., Dobres, J., Chahine, N., Mehler, B., & Reimer, B. (2020, 5). Glanceable, legible typography over complex backgrounds. *Ergonomics*, *63*(7), 864–883. Retrieved from https://doi.org/10.1080/00140139.2020.1758348 doi: 10.1080/00140139.2020.1758348

Topliss, B. H., Pampel, S. M., Burnett, G., & Gabbard, J. L. (2019, 9). Evaluating Head-Up Displays across Windshield Locations. *International Conference on Automotive User Interfaces and Interactive Vehicular Applications*, 244–253. Retrieved from https://doi.org/10.1145/3342197.3344524 doi: 10.1145/3342197.3344524

Visual Angle Calculator - Calculator Academy. (2023, 7). Retrieved from https://calculator.academy/visual-angle-calculator/

Whitney, D., & Levi, D. M. (2011, 3). Visual crowding: a fundamental limit on conscious perception and object recognition. *Trends in Cognitive Sciences*, *15*(4), 160–168. Retrieved from https://pmc.ncbi.nlm.nih.gov/articles/PMC3070834/#:~:text=vi)%20Temporal%20tuning,%2C%20%5B16%2C%2021%5D). doi: 10.1016/j.tics.2011.02.005

Wickens, C. (2015, 01). Noticing events in the visual workplace: The seev and nseev models. In (p. 749-768). doi: 10.1017/CBO9780511973017.046

Appendices Overview

Here you can find the supplementary materials for this paper:
- **Appendix A:** Consent Form
- **Appendix B:** Post Experiment Questionnaire
- **Appendix C:** MATLAB Code – Scripts used for all analyses.

A Consent Form

Visual attention: An eye-tracking experiment

Informed consent form for participants

Researchers

Khushboo Sharma, MSc student
E-mail: k.sharma-2@student.tudelft.nl

Dr.ir. Y.B. Eisma
E-mail: y.b.eisma@tudelft.nl

Location

Room F-0-640
Department of Cognitive Robotics
Faculty of Mechanical Engineering, Delft University of Technology
Mekelweg 2, 2628 CD Delft

This document describes the purpose of this study, the experimental procedure, the right to withdraw, and data handling procedures. Read all sections carefully and answer the questions on page 2.

NOTE: The measurement equipment functions better with contact lenses than glasses. If possible, please wear contact lenses instead of glasses.

Research purpose

The aim of this experiment is to investigate, by means of eye-tracking, the legibility of text on visually complex backgrounds.

Experimental procedure

Before the experiment: You will be asked to rest your head on the support and look at specific places on the screen so that we can calibrate the eye-tracking equipment.

During the experiment: First, you will be asked to locate and identify words that will appear on a visually complex background.

After the experiment: You will be asked to complete a short questionnaire about basic demographic characteristics, including gender, age, and whether you wear glasses/contacts.

Experimental setup with head support and eye-tracker.

Experiment duration

The experiment will take about 45 minutes.

Risk of participating

There are no known risks for you in this study. Some minor eyestrain or discomfort may arise from the monitoring task. If at any point you begin to feel uneasy for any reason, please do not hesitate to inform the experimenter so that you can take a break to counteract any such symptoms.

Data handling

All data in this study will be collected and stored anonymously. You will not be personally identifiable in any future publications based on this work or in any data files that may be stored in an online repository or shared with other researchers.

This signed consent form will be kept in a dedicated locker.

Right to withdraw

Your participation is completely voluntary, and you may stop at any time during the experiment for any reason. You have the right to refuse to participate or to withdraw from the experiment at any point before the end of your participation, without negative consequences and without having to provide any explanation. Please note that because the data are collected anonymously, it will not be possible to withdraw your data after your participation.

Please respond to the following statements

Statement	Yes	No
I consent to participate voluntarily in this study.	○	○
I have read and understood the information provided in this document.	○	○
I understand that I can withdraw from the study at any point before the completion of my participation, without any negative consequences.	○	○
I agree that the data collected during the experiment as described above will be used for academic research and may be anonymously presented in publications and public data repositories.	○	○

Signature

Name:

Date:

Signature: ____________________

B QUESTIONNAIRES

Post Experiment Questionnaire - Visual attention: An eye-tracking experiment

Thank you for your participation in the eye-tracking experiment. Please fill in the questionnaire below as the final step. If you have any questions or need clarification, feel free to ask the researcher.

* Required

* This form will record your name, please fill your name.

Section 1: Participant information

1

Participant ID *

2

Age *

3

Gender *

- ◯ Woman
- ◯ Man
- ◯ Non-binary
- ◯ Prefer not to say

4

Did you wear any visual aids during the experiment? *

- ◯ Yes, I wore glasses during the experiment
- ◯ Yes, I wore contact lenses during the experiment
- ◯ No, I usually wear glasses or contact lenses, but not during the experiment
- ◯ No, I usually don't wear glasses or contact lenses

5

Do you have any reading disabilities (e.g., dyslexia)? If yes, please select 'other' and specify. *

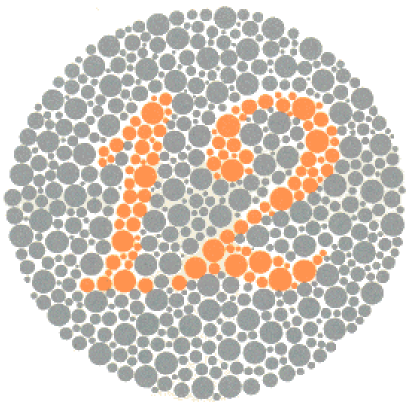
- ◯ No
- ◯ Not sure
- ◯ Other

Section 2: Color blindness check

Each of the six images below contains a circular plate made up of various coloured dots. A number can be seen in most of the plates, although in some plates, you will see nothing else than unrelated dots. For each number, type the number that you see, if any.

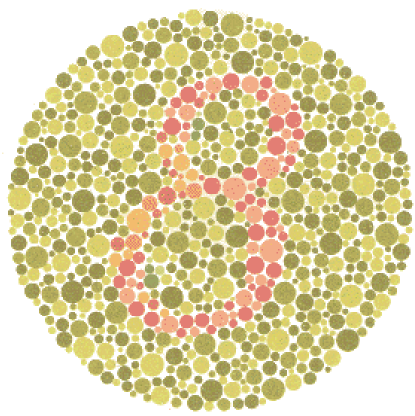
6

If you see a number in the image on the right, type it here. If you do not see any number, type 'N'. *



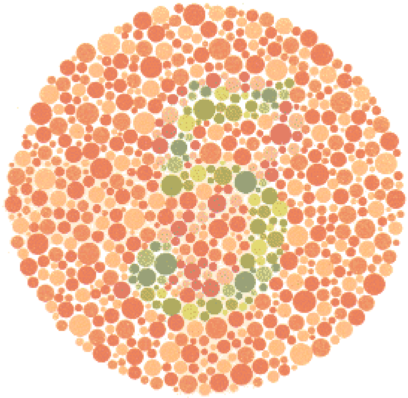
7

If you see a number in the image on the right, type it here. If you do not see any number, type 'N'. *



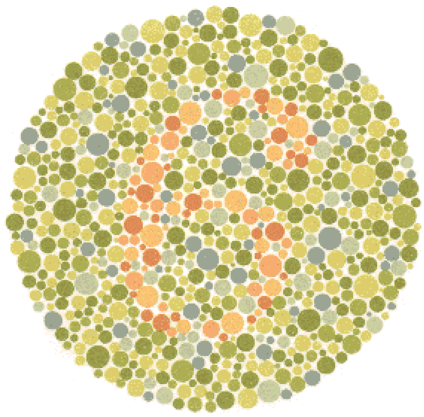
8

If you see a number in the image on the right, type it here. If you do not see any number, type 'N'. *



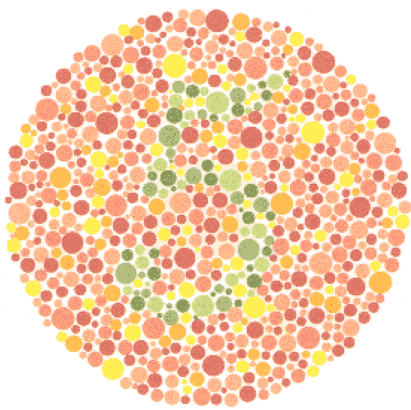
9

If you see a number in the image on the right, type it here. If you do not see any number, type 'N'. *



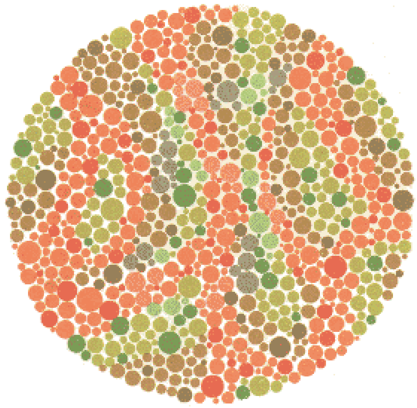
10

If you see a number in the image on the right, type it here. If you do not see any number, type 'N'. *



11

If you see a number in the image on the right, type it here. If you do not see any number, type 'N'. *



Section 3: General task experience

12

On a scale from 1 ('Not at all focused') to 5 ('Completely focused'), how focused did you feel during the task? *

	1	2	3	4	5
Focus level	○	○	○	○	○

13

Overall, how difficult was it to find the words? *

- ○ Very easy
- ○ Easy
- ○ Neutral
- ○ Difficult
- ○ Very difficult

14

Overall, how difficult was it to read the words once you found them? *

- ○ Very easy
- ○ Easy
- ○ Neutral
- ○ Difficult
- ○ Very difficult

Section 4: Perceived variation during the task

15

Did you notice differences in how easy or difficult it was to find and read the words during the experiment? *

- ◯ No, they were all equally easy/difficult to find and read
- ◯ Yes, some were slightly easier or harder to find and read
- ◯ Yes, some were much easier or harder to find and read
- ◯ Not sure / I didn’t notice

16

If you noticed differences in how easy or difficult it was to find and read the words, what do you think caused these differences? *(Select all that apply)* *

- ☐ Background visual noise
- ☐ Position of the word
- ☐ Fontsize of the word
- ☐ Color or shadow of the word
- ☐ I didn’t notice any differences
- ☐ Other

Section 5: Feedback and Impressions

17

Did anything make the task easier or harder? *

18

Was there anything about the color or shadow of the words that stood out to you, positively or negatively? *

19

Did you use any search strategies during each trial? *

20

Do you have any feedback or other remarks about the experiment? *

C MATLAB Code

```
clc; clear;

% Load fixation-annotated data
load('Grouped_EyetrackingData_FixationsTrimmedCleaned.mat', 'groupedByFontsizeAndCondition');

% Define AOI sizes per font (+30 +30) on each side
aoiSizes.f12 = [147, 91]; %87 x 31 being the max pixel sizes  % width, height in pixels
aoiSizes.f18 = [169, 96]; %109 x 36 being the max pixel sizes
aoiSizes.f24 = [192, 101]; %132 x 41 being the max pixel sizes
aoiSizes.f69 = [365, 144]; %305 x 84 being the max pixel sizes %30percent total vs 20 percent

fontsizeFields = fieldnames(groupedByFontsizeAndCondition);

% Loop through font sizes
for f = 1:numel(fontsizeFields)
    fsField = fontsizeFields{f};
    condStruct = groupedByFontsizeAndCondition.(fsField);
    aoiSize = aoiSizes.(fsField);  % Get AOI width and height for this font

    condFields = fieldnames(condStruct);

    % Loop through each condition
    for c = 1:numel(condFields)
        condField = condFields{c};
        trials = condStruct.(condField);

        for t = 1:numel(trials)
            trial = trials{t};
            if ~isfield(trial, 'Fixation_stats_struct') || isempty(trial.Fixation_stats_struct)
                fprintf(' Skipped trial %d in fontsize %s, condition %s  no fixation stats found.\n', t, fsField, condField);
                continue;
            end

            % Get stimulus position
            stimX = trial.stimulusX;
            stimY = trial.stimulusY;

            % AOI boundaries
            halfW = aoiSize(1) / 2;
            halfH = aoiSize(2) / 2;

            AOI_left   = stimX - halfW;
            AOI_right  = stimX + halfW;
            AOI_top    = stimY - halfH;
            AOI_bottom = stimY + halfH;

            % Check each fixation
            % Check each fixation in Fixation_stats_struct
            for fx = 1:numel(trial.Fixation_stats_struct)
                fix = trial.Fixation_stats_struct(fx);

                % fix.x and fix.y are assumed scalar values for this fixation
                inAOI = fix.x >= AOI_left & fix.x <= AOI_right & ...
                    fix.y >= AOI_top & fix.y <= AOI_bottom;

                % Add the inAOI field to the struct element
                trial.Fixation_stats_struct(fx).inAOI = inAOI;
            end

            % Save updated trial
            groupedByFontsizeAndCondition.(fsField).(condField){t} = trial;
        end
    end
end
```

```

% Save updated structure
save('Grouped_EyetrackingData_Fixations_WithAOI.mat', 'groupedByFontsizeAndCondition');
fprintf('   ␣Saved:␣Grouped_EyetrackingData_Fixations_WithAOI.mat\n');
```

Listing 1: MATLAB Code to Extract fixations in AOI

```
clc; clear;

% Load data
load('Grouped_EyetrackingData_Fixations_WithAOI.mat');

fontsizeLabels = {'f12', 'f18', 'f24', 'f69'};
conditionLabels = {'1S', '2S', '3S', '4S', '5S', '6S', ...
                   '1NS', '2NS', '3NS', '4NS', '5NS', '6NS'};

Results = struct();
%processingTimeResults = struct();
skippedDueToInvalidResponse = table([], [], [], ...
    'VariableNames', {'TrialID', 'ParticipantResponse', 'CorrectResponse'});


for f = 1:numel(fontsizeLabels)
    fontsize = fontsizeLabels{f};

    for c = 1:numel(conditionLabels)
        condition = conditionLabels{c};
        fieldName = ['cond_' condition];

        % Initialize result containers
        Results.(fontsize).(fieldName) = [];
        %processingTimeResults.(fontsize).(fieldName) = [];

        if ~isfield(groupedByFontsizeAndCondition.(fontsize), fieldName)
            fprintf('      ␣Skipped:␣%s␣-␣%s␣(missing␣field)\n', fontsize, condition);
            continue;
        end

        trials = groupedByFontsizeAndCondition.(fontsize).(fieldName);
        numTrials = numel(trials);
        fprintf('\ n   ␣Processing␣%d␣trials␣for␣%s␣-␣%s\n', numTrials, fontsize, condition);

        for t = 1:numTrials
            trial = trials{t};

            if ~isfield(trial, 'Fixation_stats_struct') || isempty(trial.
                Fixation_stats_struct)
                fprintf('      ␣Trial␣%d␣skipped:␣missing␣Fixation_stats_struct\n', t);
                continue;
            end

            fs = trial.Fixation_stats_struct;

            if ~isfield(fs, 'start') || ~isfield(fs, 'inAOI') || isempty(fs.start)
                fprintf('      ␣Trial␣%d␣skipped:␣missing␣"start"␣or␣"inAOI"\n', t);
                continue;
            end

            response = strtrim(lower(trial.Participant_response));
            correct = strtrim(lower(trial.correct_response));

            % Before processingTime calculation
            forceProcessingNaN = false;

            % Check for no response or 'p'
            if isempty(response) || strcmp(response, 'p')
                fprintf('      ␣Trial␣%d:␣participant␣did␣not␣see␣the␣word␣   ␣processingTime
                    ␣set␣to␣NaN,␣noticeability␣recorded\n', t);
```

```
                forceProcessingNaN = true;  % mark processing time missing, but do NOT skip trial
                % no continue; so trial proceeds
            end

            % Check for correctness with typo tolerance
            if ~isCorrectResponse(response, correct)
                fprintf('      ␣Trial␣%d:␣incorrect␣or␣invalid␣response␣   ␣processingTime␣set␣to␣NaN,␣noticeability␣recorded\n', t);

                % Capture TrialID if available; fallback to 'Unknown'
                if isfield(trial, 'TrialID')
                    trialID = trial.TrialID;
                else
                    trialID = sprintf('%s_%s_Trial%d', fontsize, condition, t);
                end

                % Append to skipped table (for your tracking)
                newRow = {trialID, response, correct};
                skippedDueToInvalidResponse = [skippedDueToInvalidResponse; newRow];

                forceProcessingNaN = true;  % mark processing time missing, but do NOT skip trial
                % no continue; so trial proceeds
            end

            % Identify last group of consecutive AOI fixations (from end)
            fixationInAOI = fs.inAOI;
            numFixations = length(fixationInAOI);

            % Reverse loop to find the last group of AOI fixations
            lastGroupStart = -1;
            lastGroupEnd = -1;

            for i = numFixations:-1:1
                if fixationInAOI(i)
                    if lastGroupEnd == -1
                        lastGroupEnd = i;  % Mark end of the group
                    end
                    lastGroupStart = i;     % Keep updating to find the start
                elseif lastGroupEnd ~= -1
                    % We just passed through the last AOI group
                    break;
                end
            end

            % Check if an AOI group was found
            if lastGroupStart == -1
                fprintf('      ␣Trial␣%d␣skipped:␣no␣fixation␣in␣AOI\n', t);
                continue;
            end

            % Get noticeability from the first fixation in the last AOI group
            timeToFirstFixation = fs.start(lastGroupStart);

            % Reaction time values
            rtVal = -1;
            if isfield(trial, 'RT') && ~isempty(trial.RT)
                rtVal = trial.RT;
            end

            rtInvVal = -1;
            if isfield(trial, 'RT_gaze_inside_invisible_boundry') && ~isempty(trial.RT_gaze_inside_invisible_boundry)
                rtInvVal = trial.RT_gaze_inside_invisible_boundry;
            end

            % Calculate processing time
```

```
            if rtInvVal ~= -1
                processingTime = rtInvVal - timeToFirstFixation;
            else
                processingTime = rtVal - timeToFirstFixation;
            end

            if (trial.timestamps(end) - trial.timestamps(1) >= 30000)
                timeToFirstFixation = 30000;
                processingTime = NaN;
            end

            % Override processingTime if flagged
            if forceProcessingNaN
                processingTime = NaN;
            end

            % Calculate trial length from timestamps (if not done yet)
            if isfield(trial, 'timestamps') && numel(trial.timestamps) >= 2
                trialLength = trial.timestamps(end) - trial.timestamps(1);
            else
                trialLength = 0;
            end

            % Determine reactionTime with your new condition
            if (trialLength >= 30000) || (rtInvVal == -1 && rtVal == -1)
                reactionTime = 30000;
            else
                if rtInvVal ~= -1
                    reactionTime = rtInvVal;
                elseif rtVal ~= -1
                    reactionTime = rtVal;
                else
                    reactionTime = NaN;  % fallback just in case
                end
            end

            % Store results
            %noticeabilityResults.(fontsize).(fieldName)(end+1) = timeToFirstFixation;
            %processingTimeResults.(fontsize).(fieldName)(end+1) = processingTime;
            participantID = trial.Session_Name_;  % or trial.Participant_ID if available
            Background = trial.Background;
            PositionX = trial.stimulusX;
            PositionY = trial.stimulusY;

            % Store results in a table or struct array
            entry = struct( ...
                'Participant', participantID, ...
                'Background', Background, ...
                'Noticeability', timeToFirstFixation, ...
                'ProcessingTime', processingTime, ...
                'ReactionTime', reactionTime, ...
                'X', PositionX, ...
                'Y', PositionY ...
                );

            if ~isfield(Results.(fontsize).(fieldName), 'data')
                Results.(fontsize).(fieldName).data = [];
            end

            Results.(fontsize).(fieldName).data = [Results.(fontsize).(fieldName).data; entry
                ];


            %fprintf('    Trial %d: Noticeability = %.2f ms | ProcessingTime = %.2f ms\n',
                ...
                    %t, timeToFirstFixation, processingTime);
        end
        % Extract numeric vectors from the struct array
```

```
        dataEntries = Results.(fontsize).(fieldName).data;

        if ~isempty(dataEntries)
            % Convert struct array to table for easy numeric access
            T = struct2table(dataEntries);

            % Only use valid numbers (exclude NaNs)
            noticeVals = T.Noticeability(~isnan(T.Noticeability));
            procVals   = T.ProcessingTime(~isnan(T.ProcessingTime));
            rtVals     = T.ReactionTime(~isnan(T.ReactionTime));

            % Compute mean and std
            meanNotice = mean(noticeVals);
            stdNotice  = std(noticeVals);

            meanProc   = mean(procVals);
            stdProc    = std(procVals);

            meanRT     = mean(rtVals);
            stdRT      = std(rtVals);

            % Print nicely
            fprintf('\ n  ␣Summary␣for␣%s␣-␣%s\n', fontsize, condition);
            fprintf('Noticeability:␣Mean␣=␣%.2f␣ms,␣Std␣=␣%.2f␣ms\n', meanNotice, stdNotice);
            fprintf('ProcessingTime:␣Mean␣=␣%.2f␣ms,␣Std␣=␣%.2f␣ms\n', meanProc, stdProc);
            fprintf('ReactionTime:␣Mean␣=␣%.2f␣ms,␣Std␣=␣%.2f␣ms\n', meanRT, stdRT);
        end

    end
end

Results.(fontsize).(fieldName).Summary = struct( ...
    'MeanNoticeability', meanNotice, ...
    'StdNoticeability', stdNotice, ...
    'MeanProcessingTime', meanProc, ...
    'StdProcessingTime', stdProc, ...
    'MeanReactionTime', meanRT, ...
    'StdReactionTime', stdRT ...
);

%% Average per participant
averagedResults = struct();

for f = 1:numel(fontsizeLabels)
    fontsize = fontsizeLabels{f};
    for c = 1:numel(conditionLabels)
        cond = conditionLabels{c};
        fieldName = ['cond_' cond];

        if isfield(Results.(fontsize), fieldName)
            data = Results.(fontsize).(fieldName).data;

            % Convert struct array to table for easy grouping
            if ~isempty(data)
                T = struct2table(data);

                % Exclude NaNs before grouping
                T = T(~isnan(T.Noticeability) & ~isnan(T.ProcessingTime) & ~isnan(T.
                    ReactionTime), :);

                % Group by Participant and average
                grouped = groupsummary(T, 'Participant', 'mean', {'Noticeability', '
                    ProcessingTime', 'ReactionTime'});

                % Save averaged results
                averagedResults.(fontsize).(fieldName).Participants = grouped.Participant;
                averagedResults.(fontsize).(fieldName).Noticeability = grouped.
                    mean_Noticeability;
```

```
                averagedResults.(fontsize).(fieldName).ProcessingTime = grouped.mean_ProcessingTime;
                averagedResults.(fontsize).(fieldName).ReactionTime   = grouped.mean_ReactionTime;
            end
        end
    end
end

fprintf('\n---␣Normality␣Check␣for␣Noticeability␣(per␣participant␣means)␣---\n');
for f = 1:numel(fontsizeLabels)
    fontsize = fontsizeLabels{f};

    for c = 1:numel(conditionLabels)
        cond = conditionLabels{c};
        fieldName = ['cond_' cond];

        if isfield(averagedResults.(fontsize), fieldName)
            values = averagedResults.(fontsize).(fieldName).Noticeability; % one value per participant

            if numel(values) > 4  % Lillietest needs a few samples
                [h,p] = lillietest(values);
                fprintf('Font:␣%-4s␣|␣Condition:␣%-8s␣|␣H=%d␣|␣p=%.4f\n', ...
                    fontsize, cond, h, p);
            end
        end
    end
end

fprintf('\n---␣Normality␣Check␣for␣Processing␣Time␣(H=1␣means␣NOT␣normal)␣---\n');

for f = 1:numel(fontsizeLabels)
    fontsize = fontsizeLabels{f};

    for c = 1:numel(conditionLabels)
        cond = conditionLabels{c};
        fieldName = ['cond_' cond];

        % Check if data exists
        if isfield(averagedResults.(fontsize), fieldName)
            values = averagedResults.(fontsize).(fieldName).ProcessingTime;

            if ~isempty(values) && numel(values) > 4  % Minimum samples for reliable test
                [h, p] = lillietest(values);
                fprintf('Font:␣%-4s␣|␣Condition:␣%-4s␣|␣H=%d␣|␣p=%.4f\n', ...
                        fontsize, cond, h, p);
            end
        end
    end
end

save('Noticeability_ProcessingTime_Results.mat', 'Results', 'averagedResults');
fprintf('\ n  ␣Results␣saved␣to␣Noticeability_ProcessingTime_Results.mat\n');
```

Listing 2: MATLAB Code Meaningful fixations and trial filtration

```
clear all; clc;

%% Typo tolerant
resultsFolder = 'C:\...\Cleaned_data\'; %% Folder path with the Results_file from Experiment Builder
fileList = dir(fullfile(resultsFolder, 'RESULTS_FILE_*.txt'));

% Prepare storage
fontsizes = [12, 18, 24, 69];
groupedByFontsize = struct();
groupedByFontsizeAndCondition = struct();
```

```

% Initialize
for f = fontsizes
    groupedByFontsize.(sprintf('f%d', f)) = [];
    groupedByFontsizeAndCondition.(sprintf('f%d', f)) = struct();
end

% Loop through all result files
for i = 1:length(fileList)
    fileName = fileList(i).name;
    filePath = fullfile(resultsFolder, fileName);


    tokens = regexp(fileName, 'RESULTS_FILE_(\d+)_(\d+)\.txt', 'tokens');
    if isempty(tokens)
        warning('Filename does not match pattern: %s', fileName);
        continue;
    end

    participantID = str2double(tokens{1}{1});  % 19
    fontsize      = str2double(tokens{1}{2});  % 12
    fsField       = sprintf('f%d', fontsize);  % "f12"


    fprintf('File: %s -> participantID=%d, fontsize=%d\n', fileName, participantID, fontsize)
        ;


    % Read table
    T = readtable(filePath, 'Delimiter', '\t');

    % Initialize correctness vector
    correct = false(height(T), 1);


    % Apply typo-tolerant accuracy check
    for r = 1:height(T)
        target = string(T.correct_response{r});
        response = string(T.Participant_response{r});

        condition = strtrim(T.Current_TransparencyCondition{r});

        % Use your isCorrectResponse function (fixed maxDistance = 2)
        correct(r) = isCorrectResponse(response, target, 2);

        % If incorrect, print details
        if ~correct(r)
            fprintf('Incorrect trial in %s (row %d): Target="%s", Response="%s", Condition=%s
                \n', ...
                fileName, r, target, response, condition);
        end

        % Store per condition per participant
        if strcmpi(condition,'9I'), continue; end  % skip 9I

        condField = matlab.lang.makeValidName(['cond_' condition]);

        % Initialize condition struct if needed
        if ~isfield(groupedByFontsizeAndCondition.(fsField), condField)
            groupedByFontsizeAndCondition.(fsField).(condField) = struct();
        end

        % Initialize participant array if needed
        participantField = ['p' num2str(participantID)];
        if ~isfield(groupedByFontsizeAndCondition.(fsField).(condField), participantField)
            groupedByFontsizeAndCondition.(fsField).(condField).(participantField) = [];
        end

```

```
        % Append this trial correctness
        groupedByFontsizeAndCondition.(fsField).(condField).(participantField) = ...
            [groupedByFontsizeAndCondition.(fsField).(condField).(participantField), correct(r)];
    end


    % Store per fontsize
    groupedByFontsize.(fsField) = [groupedByFontsize.(fsField); correct];
end

% Define custom labels for the conditions
customLabels = {'Green NS', 'Green S', 'White NS', 'White S', '30% NS', '30% S', '50% NS', '50% S', '75% NS', '75% S', '100% NS', '100% S'};  % replace with your actual labels

figure;

figure;
for i = 1:numel(fontsizes)
    f = fontsizes(i);
    fsField = sprintf('f%d', f);

    condStruct = groupedByFontsizeAndCondition.(fsField);
    condNames = fieldnames(condStruct);
    condAcc = zeros(1, numel(condNames));

    for c = 1:numel(condNames)
        % Extract participant-wise accuracies
        participantStruct = condStruct.(condNames{c});
        participantIDs = fieldnames(participantStruct);  % <-- use fieldnames here
        partAcc = zeros(1, numel(participantIDs));

        for k = 1:numel(participantIDs)
            pid = participantIDs{k};
            trials = participantStruct.(pid);
            partAcc(k) = mean(trials); % mean for this participant
        end

        % Now average across participants
        condAcc(c) = mean(partAcc) * 100;
    end

    subplot(2, 2, i);
    bar(categorical(condNames), condAcc);
    xticklabels(customLabels);  % custom x-axis labels, same for all subplots
    title(sprintf('Accuracy per Condition (Fontsize %d)', f));
    ylabel('Accuracy (%)');
    ylim([0 100]); grid on;

    % Add horizontal line at 95%
    yline(95, '--r', 'LabelHorizontalAlignment', 'left', ...
    'LabelVerticalAlignment', 'bottom', 'LineWidth', 1.5);
end
```

Listing 3: MATLAB Code Accuracy plot

```
clc; clear all;

load('Noticeability_ProcessingTime_Factors.mat', 'T'); %% Choose which factor levels to keep
selectedFonts = {'f12','f18','f24', 'f69'}; %% Choose which factor levels to kee
selectedConditions= {'cond_1','cond_2','cond_3', 'cond_4', 'cond_5', 'cond_6'};
selectedShadows = {'S', 'NS'};

% Ensure categorical variables
T.FontSize     = categorical(T.FontSize);
T.Condition    = categorical(T.Condition);
T.Shadow       = categorical(T.Shadow);
T.Participant = categorical(T.Participant); % clean participant ID
```

```
T.Background   = categorical(T.Background); % assuming background column exists

% === Filter by selected factors ===
T_filtered = T(ismember(T.FontSize, selectedFonts) & ...
               ismember(T.Condition, selectedConditions) & ...
               ismember(T.Shadow, selectedShadows), :);

% === Remove rows with NaNs in the dependent variables ===
T_clean = T_filtered(~isnan(T_filtered.ProcessingTime) & ...
                     ~isnan(T_filtered.ReactionTime) & ...
                     ~isnan(T_filtered.Noticeability), :);

% === Drop unused factor levels ===
T_clean.FontSize  = removecats(T_clean.FontSize);
T_clean.Condition = removecats(T_clean.Condition);
T_clean.Shadow    = removecats(T_clean.Shadow);
T_clean.Participant = removecats(T_clean.Participant);
T_clean.Background  = removecats(T_clean.Background);

% Convert positions from cell to numeric
T_clean.Xposition = cell2mat(T_clean.Xposition);
T_clean.Yposition = cell2mat(T_clean.Yposition);

% Define stimulus area bounds
H_min = 202; H_max = 1717;   % horizontal
V_min = 62;  V_max = 1018;   % vertical

% Normalize positions
T_clean.HorNorm = (T_clean.Xposition - H_min) / (H_max - H_min);
T_clean.VerNorm = (T_clean.Yposition - V_min) / (V_max - V_min);

% Replace negative ProcessingTime values with NaN
negProcessIdx = T_clean.ProcessingTime < 0;
if any(negProcessIdx)
    fprintf('Setting␣%d␣negative␣ProcessingTime␣values␣to␣NaN.\n', sum(negProcessIdx));
    T_clean.ProcessingTime(negProcessIdx) = NaN;
end

%Noticeability
glme_notice = fitglme(T_clean, 'Noticeability␣~␣FontSize*Condition*Shadow␣+␣HorNorm␣+␣VerNorm␣+␣(1|Participant)␣+␣(1␣+␣Condition|Background)', 'Distribution','Gamma','Link','log');
anova(glme_notice)   % fixed effects
%disp(glme_notice)    % model summary
meanValue = mean(T_clean.Noticeability);
stdValue = std(T_clean.Noticeability);
minValue = min(T_clean.Noticeability);
maxValue = max(T_clean.Noticeability);

fprintf('Mean:␣%.2f,␣SD:␣%.2f,␣Min:␣%.2f,␣Max:␣%.2f\n', meanValue, stdValue, minValue, maxValue);

%Processingtime
glme_time = fitglme(T_clean, ...
    'ProcessingTime␣~␣FontSize*Condition*Shadow␣+␣HorNorm␣+␣VerNorm␣+␣(1|Participant)␣+␣(1␣+␣FontSize|Background)', ...
    'Distribution', 'Gamma', 'Link', 'log'); %(1 + FontSize|Background)
anova(glme_time)
%disp(glme_time)

%% Descriptive statistics
summaryVars = {'Noticeability','ProcessingTime'};

%% 1. Table with one row per FontSize (collapsed over Condition and Shadow)
summaryFont_Mean = varfun(@mean, T_clean, ...
    'InputVariables', summaryVars, ...
    'GroupingVariables', {'FontSize'});

summaryFont_SD = varfun(@std, T_clean, ...
```

```
    'InputVariables', summaryVars, ...
    'GroupingVariables', {'FontSize'});

% Merge mean and SD
summaryFont = join(summaryFont_Mean, summaryFont_SD, 'Keys', {'FontSize'});

% Rename columns
summaryFont.Properties.VariableNames{'mean_Noticeability'} = 'Noticeability_Mean';
summaryFont.Properties.VariableNames{'std_Noticeability'}  = 'Noticeability_SD';
summaryFont.Properties.VariableNames{'mean_ProcessingTime'} = 'ProcessingTime_Mean';
summaryFont.Properties.VariableNames{'std_ProcessingTime'}  = 'ProcessingTime_SD';

disp('Summary␣table:␣one␣row␣per␣FontSize')
disp(summaryFont)

% 2. Table with one row per Condition (collapsed over FontSize and Shadow)
summaryCond_Mean = varfun(@mean, T_clean, ...
    'InputVariables', summaryVars, ...
    'GroupingVariables', {'Condition'});

summaryCond_SD = varfun(@std, T_clean, ...
    'InputVariables', summaryVars, ...
    'GroupingVariables', {'Condition'});

% Merge mean and SD
summaryCond = join(summaryCond_Mean, summaryCond_SD, 'Keys', {'Condition'});

% Rename columns
summaryCond.Properties.VariableNames{'mean_Noticeability'} = 'Noticeability_Mean';
summaryCond.Properties.VariableNames{'std_Noticeability'}  = 'Noticeability_SD';
summaryCond.Properties.VariableNames{'mean_ProcessingTime'} = 'ProcessingTime_Mean';
summaryCond.Properties.VariableNames{'std_ProcessingTime'}  = 'ProcessingTime_SD';

disp('Summary␣table:␣one␣row␣per␣Condition')
disp(summaryCond)


%% Loop over conditions and plot mean per font
fontSizesDeg = [0.1, 0.15, 0.2, 0.6];  % X-axis labels
for j = 1:numel(conditions)
    cond = conditions(j);
    meanNotice = zeros(1,numel(fontSizesDeg));
    for i = 1:numel(fontSizesDeg)
        % Index for this FontSize x Condition combination
        idx = (FontDeg == fontSizesDeg(i)) & (T_clean.Condition == cond);
        meanNotice(i) = mean(T_clean.Noticeability(idx));
    end
    % Plot line for this condition
    plot(fontSizesDeg, meanNotice, '-o', 'Color', colors(j,:), 'Marker', markerStyles{mod(j
        -1,numel(markerStyles))+1}, 'LineWidth', 1.5);
end



% Map FontSize categorical to visual angles
fontMapping = containers.Map({'f12','f18','f24','f69'}, [0.1, 0.15, 0.2, 0.6]);
FontDeg = zeros(height(T_clean),1);
for k = 1:height(T_clean)
    FontDeg(k) = fontMapping(char(T_clean.FontSize(k)));
end

% Conditions (categorical or numeric)
conditions = unique(T_clean.Condition);

% Define your custom legend labels here
customLegend = {'Plain␣green␣text', 'Plain␣white␣text', '30%␣billboard', ...
                '50%␣billboard', '75%␣billboard', '100%␣billboard'};
```

```
% Prepare for plotting
colors = lines(numel(conditions)); % assign colors to conditions
markerStyles = {'o','s','^','d','v','>'}; % marker styles

%% NOTICEABILITY PLOT
figure; hold on;
fontSizesDeg = [0.1, 0.15, 0.2, 0.6];  % X-axis labels
for j = 1:numel(conditions)
    cond = conditions(j);
    meanNotice = zeros(1,numel(fontSizesDeg));
    for i = 1:numel(fontSizesDeg)
        % Index for this FontSize x Condition combination
        idx = (FontDeg == fontSizesDeg(i)) & (T_clean.Condition == cond);
        meanNotice(i) = mean(T_clean.Noticeability(idx), 'omitnan');
    end
    % Plot line for this condition
    plot(fontSizesDeg, meanNotice, '-o', 'Color', colors(j,:), ...
        'Marker', markerStyles{mod(j-1,numel(markerStyles))+1}, 'LineWidth', 1.5);
end

xlabel('Font␣size␣(  )');
ylabel('Mean␣Noticeability␣(ms)');
xlim([0.05 0.65]);
xticks(fontSizesDeg);
legend(customLegend, 'Location', 'best');
title('Noticeability␣vs␣Font␣Size␣per␣Condition');
grid on; box on;

%% PROCESSING TIME PLOT
figure; hold on;
for j = 1:numel(conditions)
    cond = conditions(j);
    meanProc = zeros(1,numel(fontSizesDeg));
    for i = 1:numel(fontSizesDeg)
        idx = (FontDeg == fontSizesDeg(i)) & (T_clean.Condition == cond);
        meanProc(i) = mean(T_clean.ProcessingTime(idx), 'omitnan');
    end
    plot(fontSizesDeg, meanProc, '-o', 'Color', colors(j,:), ...
        'Marker', markerStyles{mod(j-1,numel(markerStyles))+1}, 'LineWidth', 1.5);
end

xlabel('Font␣size␣(  )');
ylabel('Mean␣Processing␣Time␣(ms)');
xlim([0.05 0.65]);
xticks(fontSizesDeg);
legend(customLegend, 'Location', 'best');
title('Processing␣Time␣vs␣Font␣Size␣per␣Condition');
grid on; box on;



%% Post-hoc comparison

DV = T_clean.ProcessingTime;         % or Noticeability
Subject = T_clean.Participant;       % for within-subject pairing
Font = T_clean.FontSize;
Cond = T_clean.Condition;
Shadow = T_clean.Shadow;
glme = glme_time;                    % use for Processingtime
%glme = glme_notice;                 % use for Noticeability

all_pvals_posthoc = [];
all_pairs_posthoc = {};

%% FontSize
uniqueFont = categories(Font);
pFont = anova(glme);  % p-value for FontSize from GLME
pFont = pFont.pValue(strcmp(pFont.Term,'FontSize'));
```

```

if length(uniqueFont) > 2 %&& pFont < 0.05
    dataTable = table(Font, DV, Subject, 'VariableNames', {'Font','DV','Participant'});
    [pvals, pairs] = runWilcoxonPosthoc(dataTable, 'Font');
    all_pvals_posthoc = [all_pvals_posthoc, pvals];
    all_pairs_posthoc = [all_pairs_posthoc, pairs];
else
    fprintf('FontSize␣has␣2␣or␣fewer␣levels,␣skipping␣Wilcoxon␣post-hoc.\n');
end

%% Repeat for Condition
uniqueCond = categories(Cond);
pCond = anova(glme);  % adjust to extract Condition p-value
pCond = pCond.pValue(strcmp(pCond.Term,'Condition'));

if length(uniqueCond) > 2 %&& pCond < 0.05
    dataTable = table(Cond, DV, Subject, 'VariableNames', {'Condition','DV','Participant'});
    [pvals, pairs] = runWilcoxonPosthoc(dataTable, 'Condition');
    all_pvals_posthoc = [all_pvals_posthoc, pvals];
    all_pairs_posthoc = [all_pairs_posthoc, pairs];
end

% Repeat for Shadow if needed
uniqueShadow = categories(Shadow);
pShadow = anova(glme);  % adjust to extract Condition p-value
pShadow = pShadow.pValue(strcmp(pShadow.Term,'Shadow'));

if length(uniqueShadow) > 2 && pCond < 0.05
    dataTable = table(Font, DV, Subject, 'VariableNames', {'Font','DV','Participant'});
    [pvals, pairs] = runWilcoxonPosthoc(dataTable, 'Font');
    all_pvals_posthoc = [all_pvals_posthoc, pvals];
    all_pairs_posthoc = [all_pairs_posthoc, pairs];
end

fprintf('===␣Wilcoxon␣Signed-Rank␣Post-hoc␣Results␣===\n');
for k = 1:length(all_pvals_posthoc)
    fprintf('%s:␣p␣=␣%.6f\n', all_pairs_posthoc{k}, all_pvals_posthoc(k));
end


if ~isempty(all_pvals_posthoc)
    adj_pvals = holm_bonferroni(all_pvals_posthoc);  % or FDR
    for k = 1:length(all_pvals_posthoc)
        fprintf('%s:␣raw␣p␣=␣%.5f,␣corrected␣p␣=␣%.5f\n', all_pairs_posthoc{k}, all_pvals_posthoc(k), adj_pvals(k));
    end
end
```

Listing 4: MATLAB Code for statistical analysis Font size x Condition x Shadow

```
clc; clear all;

load('Noticeability_ProcessingTime_Factors.mat', 'T'); %% Choose which factor levels to keep
selectedFonts = {'f12','f18','f24', 'f69'}; %% Choose which factor levels to kee
selectedConditions= {'cond_1','cond_2','cond_3', 'cond_4', 'cond_5', 'cond_6'};
selectedShadows = {'S', 'NS'};

% Ensure categorical variables
T.FontSize     = categorical(T.FontSize);
T.Condition    = categorical(T.Condition);
T.Shadow       = categorical(T.Shadow);
T.Participant = categorical(T.Participant); % clean participant ID
T.Background   = categorical(T.Background); % assuming background column exists

% === Filter by selected factors ===
T_filtered = T(ismember(T.FontSize, selectedFonts) & ...
               ismember(T.Condition, selectedConditions) & ...
               ismember(T.Shadow, selectedShadows), :);
```

```

% === Remove rows with NaNs in the dependent variables ===
T_clean = T_filtered(~isnan(T_filtered.ProcessingTime) & ...
                     ~isnan(T_filtered.ReactionTime) & ...
                     ~isnan(T_filtered.Noticeability), :);

% === Drop unused factor levels ===
T_clean.FontSize  = removecats(T_clean.FontSize);
T_clean.Condition = removecats(T_clean.Condition);
T_clean.Shadow    = removecats(T_clean.Shadow);
T_clean.Participant = removecats(T_clean.Participant);
T_clean.Background  = removecats(T_clean.Background);

% Convert positions from cell to numeric
T_clean.Xposition = cell2mat(T_clean.Xposition);
T_clean.Yposition = cell2mat(T_clean.Yposition);

% Define stimulus area bounds
H_min = 202; H_max = 1717;   % horizontal
V_min = 62;  V_max = 1018;   % vertical

% Normalize positions
T_clean.HorNorm = (T_clean.Xposition - H_min) / (H_max - H_min);
T_clean.VerNorm = (T_clean.Yposition - V_min) / (V_max - V_min);

% Replace negative ProcessingTime values with NaN
negProcessIdx = T_clean.ProcessingTime < 0;
if any(negProcessIdx)
    fprintf(’Setting␣%d␣negative␣ProcessingTime␣values␣to␣NaN.\n’, sum(negProcessIdx));
    T_clean.ProcessingTime(negProcessIdx) = NaN;
end

%Noticeability
glme_notice = fitglme(T_clean, ’Noticeability␣~␣FontSize*Condition*Shadow␣+␣HorNorm␣+␣VerNorm␣+␣(1|Participant)␣+␣(1␣+␣Condition|Background)’, ’Distribution’,’Gamma’,’Link’,’log’);
anova(glme_notice)   % fixed effects
%disp(glme_notice)    % model summary
meanValue = mean(T_clean.Noticeability);
stdValue = std(T_clean.Noticeability);
minValue = min(T_clean.Noticeability);
maxValue = max(T_clean.Noticeability);

fprintf(’Mean:␣%.2f,␣SD:␣%.2f,␣Min:␣%.2f,␣Max:␣%.2f\n’, meanValue, stdValue, minValue, maxValue);

%Processingtime
glme_time = fitglme(T_clean, ...
    ’ProcessingTime␣~␣FontSize*Condition*Shadow␣+␣HorNorm␣+␣VerNorm␣+␣(1|Participant)␣+␣(1␣+␣FontSize|Background)’, ...
    ’Distribution’, ’Gamma’, ’Link’, ’log’); %(1 + FontSize|Background)
anova(glme_time)
%disp(glme_time)

%% Descriptive statistics
summaryVars = {’Noticeability’,’ProcessingTime’};

%% 1. Table with one row per FontSize (collapsed over Condition and Shadow)
summaryFont_Mean = varfun(@mean, T_clean, ...
    ’InputVariables’, summaryVars, ...
    ’GroupingVariables’, {’FontSize’});

summaryFont_SD = varfun(@std, T_clean, ...
    ’InputVariables’, summaryVars, ...
    ’GroupingVariables’, {’FontSize’});

% Merge mean and SD
summaryFont = join(summaryFont_Mean, summaryFont_SD, ’Keys’, {’FontSize’});

```

```
% Rename columns
summaryFont.Properties.VariableNames{'mean_Noticeability'} = 'Noticeability_Mean';
summaryFont.Properties.VariableNames{'std_Noticeability'}  = 'Noticeability_SD';
summaryFont.Properties.VariableNames{'mean_ProcessingTime'} = 'ProcessingTime_Mean';
summaryFont.Properties.VariableNames{'std_ProcessingTime'}  = 'ProcessingTime_SD';

disp('Summary table: one row per FontSize')
disp(summaryFont)

% 2. Table with one row per Condition (collapsed over FontSize and Shadow)
summaryCond_Mean = varfun(@mean, T_clean, ...
    'InputVariables', summaryVars, ...
    'GroupingVariables', {'Condition'});

summaryCond_SD = varfun(@std, T_clean, ...
    'InputVariables', summaryVars, ...
    'GroupingVariables', {'Condition'});

% Merge mean and SD
summaryCond = join(summaryCond_Mean, summaryCond_SD, 'Keys', {'Condition'});

% Rename columns
summaryCond.Properties.VariableNames{'mean_Noticeability'} = 'Noticeability_Mean';
summaryCond.Properties.VariableNames{'std_Noticeability'}  = 'Noticeability_SD';
summaryCond.Properties.VariableNames{'mean_ProcessingTime'} = 'ProcessingTime_Mean';
summaryCond.Properties.VariableNames{'std_ProcessingTime'}  = 'ProcessingTime_SD';

disp('Summary table: one row per Condition')
disp(summaryCond)


%% Loop over conditions and plot mean per font
fontSizesDeg = [0.1, 0.15, 0.2, 0.6];  % X-axis labels
for j = 1:numel(conditions)
    cond = conditions(j);
    meanNotice = zeros(1,numel(fontSizesDeg));
    for i = 1:numel(fontSizesDeg)
        % Index for this FontSize x Condition combination
        idx = (FontDeg == fontSizesDeg(i)) & (T_clean.Condition == cond);
        meanNotice(i) = mean(T_clean.Noticeability(idx));
    end
    % Plot line for this condition
    plot(fontSizesDeg, meanNotice, '-o', 'Color', colors(j,:), 'Marker', markerStyles{mod(j-1,numel(markerStyles))+1}, 'LineWidth', 1.5);
end



% Map FontSize categorical to visual angles
fontMapping = containers.Map({'f12','f18','f24','f69'}, [0.1, 0.15, 0.2, 0.6]);
FontDeg = zeros(height(T_clean),1);
for k = 1:height(T_clean)
    FontDeg(k) = fontMapping(char(T_clean.FontSize(k)));
end

% Conditions (categorical or numeric)
conditions = unique(T_clean.Condition);

% Define your custom legend labels here
customLegend = {'Plain green text', 'Plain white text', '30% billboard', ...
                '50% billboard', '75% billboard', '100% billboard'};

% Prepare for plotting
colors = lines(numel(conditions)); % assign colors to conditions
markerStyles = {'o','s','^','d','v','>'}; % marker styles

%% NOTICEABILITY PLOT
figure; hold on;
```

```
fontSizesDeg = [0.1, 0.15, 0.2, 0.6];  % X-axis labels
for j = 1:numel(conditions)
    cond = conditions(j);
    meanNotice = zeros(1,numel(fontSizesDeg));
    for i = 1:numel(fontSizesDeg)
        % Index for this FontSize x Condition combination
        idx = (FontDeg == fontSizesDeg(i)) & (T_clean.Condition == cond);
        meanNotice(i) = mean(T_clean.Noticeability(idx), 'omitnan');
    end
    % Plot line for this condition
    plot(fontSizesDeg, meanNotice, '-o', 'Color', colors(j,:), ...
        'Marker', markerStyles{mod(j-1,numel(markerStyles))+1}, 'LineWidth', 1.5);
end

xlabel('Font␣size␣(  )');
ylabel('Mean␣Noticeability␣(ms)');
xlim([0.05 0.65]);
xticks(fontSizesDeg);
legend(customLegend, 'Location', 'best');
title('Noticeability␣vs␣Font␣Size␣per␣Condition');
grid on; box on;

%% PROCESSING TIME PLOT
figure; hold on;
for j = 1:numel(conditions)
    cond = conditions(j);
    meanProc = zeros(1,numel(fontSizesDeg));
    for i = 1:numel(fontSizesDeg)
        idx = (FontDeg == fontSizesDeg(i)) & (T_clean.Condition == cond);
        meanProc(i) = mean(T_clean.ProcessingTime(idx), 'omitnan');
    end
    plot(fontSizesDeg, meanProc, '-o', 'Color', colors(j,:), ...
        'Marker', markerStyles{mod(j-1,numel(markerStyles))+1}, 'LineWidth', 1.5);
end

xlabel('Font␣size␣(  )');
ylabel('Mean␣Processing␣Time␣(ms)');
xlim([0.05 0.65]);
xticks(fontSizesDeg);
legend(customLegend, 'Location', 'best');
title('Processing␣Time␣vs␣Font␣Size␣per␣Condition');
grid on; box on;



%% Post-hoc comparison

DV = T_clean.ProcessingTime;        % or Noticeability
Subject = T_clean.Participant;      % for within-subject pairing
Font = T_clean.FontSize;
Cond = T_clean.Condition;
Shadow = T_clean.Shadow;
glme = glme_time;                   % use for Processingtime
%glme = glme_notice;                % use for Noticeability

all_pvals_posthoc = [];
all_pairs_posthoc = {};

%% FontSize
uniqueFont = categories(Font);
pFont = anova(glme);  % p-value for FontSize from GLME
pFont = pFont.pValue(strcmp(pFont.Term,'FontSize'));

if length(uniqueFont) > 2 %&& pFont < 0.05
    dataTable = table(Font, DV, Subject, 'VariableNames', {'Font','DV','Participant'});
    [pvals, pairs] = runWilcoxonPosthoc(dataTable, 'Font');
    all_pvals_posthoc = [all_pvals_posthoc, pvals];
    all_pairs_posthoc = [all_pairs_posthoc, pairs];
```

```
else
    fprintf('FontSize␣has␣2␣or␣fewer␣levels,␣skipping␣Wilcoxon␣post-hoc.\n');
end

%% Repeat for Condition
uniqueCond = categories(Cond);
pCond = anova(glme);  % adjust to extract Condition p-value
pCond = pCond.pValue(strcmp(pCond.Term,'Condition'));

if length(uniqueCond) > 2 %&& pCond < 0.05
    dataTable = table(Cond, DV, Subject, 'VariableNames', {'Condition','DV','Participant'});
    [pvals, pairs] = runWilcoxonPosthoc(dataTable, 'Condition');
    all_pvals_posthoc = [all_pvals_posthoc, pvals];
    all_pairs_posthoc = [all_pairs_posthoc, pairs];
end

% Repeat for Shadow if needed
uniqueShadow = categories(Shadow);
pShadow = anova(glme);  % adjust to extract Condition p-value
pShadow = pShadow.pValue(strcmp(pShadow.Term,'Shadow'));

if length(uniqueShadow) > 2 && pCond < 0.05
    dataTable = table(Font, DV, Subject, 'VariableNames', {'Font','DV','Participant'});
    [pvals, pairs] = runWilcoxonPosthoc(dataTable, 'Font');
    all_pvals_posthoc = [all_pvals_posthoc, pvals];
    all_pairs_posthoc = [all_pairs_posthoc, pairs];
end

fprintf('===␣Wilcoxon␣Signed-Rank␣Post-hoc␣Results␣===\n');
for k = 1:length(all_pvals_posthoc)
    fprintf('%s:␣p␣=␣%.6f\n', all_pairs_posthoc{k}, all_pvals_posthoc(k));
end


if ~isempty(all_pvals_posthoc)
    adj_pvals = holm_bonferroni(all_pvals_posthoc);  % or FDR
    for k = 1:length(all_pvals_posthoc)
        fprintf('%s:␣raw␣p␣=␣%.5f,␣corrected␣p␣=␣%.5f\n', all_pairs_posthoc{k},
            all_pvals_posthoc(k), adj_pvals(k));
    end
end
```

Listing 5: MATLAB Code Background x Font size

```
function isCorrect = isCorrectResponse(response, correct, maxDistance)

    if nargin < 3
        maxDistance = 2;
    end

    % Convert strings to char arrays (for compatibility)
    if isstring(response), response = char(response); end
    if isstring(correct), correct = char(correct); end

    % Normalize: lowercase, trim whitespace
    response = lower(strtrim(response));
    correct  = lower(strtrim(correct));

    % Special cases: empty or 'p' = participant gave up
    if isempty(response) || strcmp(response, 'p')
        isCorrect = false;
        return;
    end

    % Exact match
    if strcmp(response, correct)
        isCorrect = true;
        return;
```

```
    end

    % Levenshtein distance
    distance = levenshtein(response, correct);
    isCorrect = distance <= maxDistance;
end
```

Listing 6: MATLAB function for typo tolerance

```
function d = levenshtein(s, t)
% levenshtein - Compute Levenshtein distance between two char arrays

    s = char(s);
    t = char(t);

    m = length(s);
    n = length(t);
    D = zeros(m+1, n+1);

    for i = 1:m+1
        D(i,1) = i-1;
    end
    for j = 1:n+1
        D(1,j) = j-1;
    end

    for i = 2:m+1
        for j = 2:n+1
            cost = ~(s(i-1) == t(j-1));  % 0 if same, 1 if different
            D(i,j) = min([
                D(i-1,j) + 1,      % deletion
                D(i,j-1) + 1,      % insertion
                D(i-1,j-1) + cost  % substitution
            ]);
        end
    end

    d = D(m+1, n+1);
end
```

Listing 7: MATLAB function for levenshtein 6

```
function adj_p = holm_bonferroni(pvals)
    % Holm-Bonferroni correction
    [p_sorted, sortIdx] = sort(pvals);
    n = length(pvals);
    adj = zeros(size(pvals));
    for i = 1:n
        adj(i) = min(1, (n - i + 1) * p_sorted(i));
    end
    % Ensure monotonicity
    for i = 2:n
        adj(i) = max(adj(i), adj(i-1));
    end
    % Return adjusted p-values in original order
    adj_p = zeros(size(pvals));
    adj_p(sortIdx) = adj;
end
```

Listing 8: My MATLAB Code for Holm-Bonferroni correction