USAARL-TECH-FR--2021-13

Visual Performance and Marksmanship Under Conditions of Glare and a Sub-Pixel Rendering Method for Generating Vernier Acuity Targets

Kevin O’Brien

Notice

Qualified Requesters

Qualified requesters may obtain copies from the Defense Technical Information Center (DTIC), Fort Belvoir, Virginia 22060. Orders will be expedited if placed through the librarian or other person designated to request documents from DTIC.

Change of Address

Organizations receiving reports from the U.S. Army Aeromedical Research Laboratory on automatic mailing lists should confirm correct address when corresponding about laboratory reports.

Disposition

Destroy this document when it is no longer needed. Do not return it to the originator.

Disclaimer

The views, opinions, and/or findings contained in this report are those of the author(s) and should not be construed as an official Department of the Army position, policy, or decision, unless so designated by other official documentation. Citation of trade names in this report does not constitute an official Department of the Army endorsement or approval of the use of such commercial items.

IRB Approval

Log Number M-10699

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
24-05-2021	Final	Oct 2017 - Present

4. TITLE AND SUBTITLE

Visual Performance and Marksmanship under Conditions of Glare and a Sub-Pixel Rendering Method for Generating Vernier Acuity Targets

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6.2 (FY18) and 6.3 (FY 19)

6. AUTHOR(S)

O'Brien, K.

5d. PROJECT NUMBER

20930

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

U.S. Army Aeromedical Research Laboratory
P.O. Box 620577
Fort Rucker, AL 36362

8. PERFORMING ORGANIZATION REPORT NUMBER

USAARL-TECH-FR-2021-13

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

United States Army Medical Research and Development Command
Military Operational Medical Research Program
504 Scott St.
Fort Detrick, MD 21702-5012

10. SPONSOR/MONITOR'S ACRONYM(S)

USAMRDC

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT

DISTRIBUTION STATEMENT A. Approved for public release; distribution unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Glare occurs when bright light scatters within the eye and reduces visibility of a target. This has been a known issue in military marksmanship for over a century. While many glare countermeasures related to marksmanship have been patented over the years, quantitative assessment of the impact of glare on marksmanship is conspicuously absent. To remedy this gap in the literature, novel instrumentation was created for psychophysical assessment of the impact of glare on marksmanship performance. Limitations in the experimental design lead to an early termination of human subjects data collection, but scientific findings as well as descriptions of tools developed for this research effort provide guidance for further empirical study. Additionally, a description of and source code for a sub-pixel rendering method of producing Vernier acuity targets is included.

15. SUBJECT TERMS

glare, marksmanship, vision, contrast, acuity, Vernier acuity, discomfort

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Loraine St. Onge, PhD
UNCLAS	UNCLAS	UNCLAS	SAR	27	19b. TELEPHONE NUMBER *(Include area code)* 334-255-6906

This page is intentionally blank.

Table of Contents

List of Figures

This page is intentionally blank.

Background

Glare occurs when a bright light causes intraocular scatter and reduces the ability to resolve detail in a viewed target. The most commonly experienced source of glare is sunlight, but artificial lighting is increasingly becoming a common source of glare at night. Shorter wavelengths (e.g., “blue” or “violet” light) cause more intraocular scatter than longer wavelengths (e.g., “red” light). Sunlight and many modern outdoor sources of artificial light (such as white LEDs or xenon bulbs) have a significant short-wave component that contributes to the production of glare.

Glare has been known to be a nuisance during U.S. Army marksmanship tasks for well over a century (Banister, 1893). The angled grooves cut along many firearms between the front and rear iron sights exist as a glare reduction measure, and there are numerous patents for glare mitigation apparatuses in the design of firearms, riflescopes, shooting glasses, and other equipment related to marksmanship. However, a search of the literature did not reveal any publications demonstrating glare to have a negative impact on marksmanship performance or quantification of such a relationship.

Methods

Macular pigment optical density (MPOD) was assessed using heterochromatic flicker photometry (HFP). Subject age was entered into the densitometer and a quick critical flicker fusion (CFF) threshold estimate was performed with a series of ascending frequency presentations. The densitometer has a built-in method for selecting a HFP flicker frequency based on age and CFF threshold, with final refinement of this frequency performed by the experimenter using feedback from the subject. Once the needed flicker frequency is established, alternating ascending and descending trials use the method of limits to identify a threshold estimate for the HFP task. By performing this HFP assessment at two retinal locations – one centrally where MPOD is highest and one peripherally where MPOD is negligible – the device is able to estimate the amount of optical filtration required to account for the sensitivity difference and then calculates a central MPOD estimate.

Custom equipment and software were developed (see Materials section) to conduct a visual performance battery. This battery was developed based on the guidance of several senior vision researchers (most of whom conspicuously retired between the development of the proposal and the start of data collection). The two initially planned assessments were MPOD measurement and marksmanship performance under various glare conditions. Visual acuity and contrast sensitivity assessments were also added because those visual capabilities are assumed essential for target identification in glare conditions. Because marksmanship with iron sights involves aligning a post or blade with a target, Vernier acuity was added to isolate the visual aspect of fine aim. A subjective discomfort assessment was also added on the assumption that disturbances in marksmanship performance may be influenced by unintended motor responses to physical discomfort.

Visual acuity was measured using logMAR charts with Landolt C optotypes of decreasing size arranged in a triangular pattern. Acuity thresholds were assessed in the standard fashion (Bailey & Lovie, 1976) based on the smallest line with a correct response and the

number of correct responses on that line. Contrast sensitivity was measured with a custom chart with Landolt C optotypes of the same size but the optotypes became lighter (i.e., less pigmented) as they descended down the chart, rather than becoming smaller.

The brightness of the glare sources was digitally controllable using pulse width modulation (PWM). With PWM, the light emitting diodes (LEDs) are turned on and off at a high frequency to create intermediate brightness values by changing the duty cycle. As the glare sources were created for this experiment and prior human subject testing data were not available, glare settings were selected a priori. Brightness could be set via PWM to any of 4096 levels (ranging from 0, which is completely off, to 4095, which is continuously on) so powers of 2 were selected for testing (i.e., 0, 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, and 4095).

Additionally, because the LEDs in the glare sources were individually adjustable, glare could be made to come from a specific direction. "Up," "down," "left," "right," and "uniform" conditions were created by turning on the top, bottom, left, and right quarter of LEDs or all LEDs. Because the directional conditions used a quarter as many LEDs, isoluminance could be maintained between the directional conditions and the uniform condition by cutting the PWM setting in the uniform condition by 75%. This was of particular interest with regard to the marksmanship task, as directional shifts in point of impact would be of interest if they were readily predictable.

Materials

A 10-meter indoor air rifle range previously constructed at the U.S. Army Aeromedical Research Laboratory (USAARL) was the catalyst for the development of this project. Pre-charged pneumatic .177 caliber air rifles (Anschutz 8001 AiR-15 models) designed to mimic an M16A2 were used.

Two custom glare generating light sources were developed for this experiment. The circuit design and board layouts were conducted at USAARL, the boards were fabricated by an outside vendor, and the component part population and soldering was performed at USAARL. Additionally, all necessary software and firmware development was performed at USAARL. The "distal source" was designed to mount around all four corners of a vision testing or marksmanship target. The "proximal source" was designed to sit approximately 1 inch in front of the subject's eye, whether mounted on a post (for the visual performance assessments) or mounted on an air rifle (for the marksmanship assessments). Note that technical details regarding these glare sources are not given here since, during the writing of this document, one of the glare sources is awaiting a patent determination.

MPOD measurements were performed using a Macular Metrics II™ densitometer. This device uses heterochromatic flicker photometry to estimate MPOD. By assessing relative sensitivity between a light source attenuated by macular pigment and a light source which is not attenuated by macular pigment at a retinal position without macular pigment and in the central visual field (where macular pigment density is the highest), the apparatus provides a non-invasive estimate of MPOD.

Visual acuity and contrast sensitivity were assessed using printed charts. These charts

were printed from vector graphic files that were procedurally generated by custom Python scripts. The targets were Landolt C's with the open gap in one of eight orientations corresponding to 45° increments. Verbal subject responses were recorded by an experimenter using a custom Python script developed for this experiment. For the visual acuity measurements, this script calculated logMAR scores of acuity.

A custom Python script was also used for recording subjective glare discomfort ratings, which were similarly given verbally by the subject but ranged from zero to ten.

Vernier acuity was assessed using a combination of a high-resolution monitor, a custom Python script (see appendix) which performs sub-pixel rendering to triple the adjustment precision, and a mirror. The sub-pixel rendering approach reduced the needed viewing distance by two thirds compared to conventional approaches of generating Vernier acuity targets on a monitor. In this process, a Vernier acuity target is drawn using the individual red, green, and blue channels ("sub-pixels") within each pixel to triple the number of available positions and effectively triple the adjustment resolution. The software was configured to generate Vernier targets with an algorithmically determined lateral offset magnitude, but a randomly assigned offset direction. If a subject correctly identified the offset direction in a forced choice task, the target would disappear and then be redrawn after a brief delay (to minimize temporal contrast effects) with a smaller lateral offset magnitude (and the offset direction again randomly assigned). If the offset direction was incorrectly identified, the subsequent trial would have a larger offset. Exit conditions were three incorrect responses or a correct response with a single sub-pixel of offset.

Results

The proximal glare source generated adequate veiling glare to create significant visual disruption. The distal glare source, even at maximum intensity, did not adequately obscure targets. This was unsurprising, given that the distal glare source was several meters away from the observer, but it was created and preliminary testing was performed in due diligence at the insistence of a senior vision scientist.

Data collection was halted after the first subject due to several issues. During zero and low intensity glare conditions, the first subject's visual acuity exceeded 0.0 logMAR (which is analogous to "20/20" in Snellen acuity). As might be expected in that scenario, Vernier acuity thresholds exceeded measurement capability, as a correct forced choice response to a single sub-pixel offset condition terminates the trial in the software. For all visual performance measures, threshold estimates were relatively stable (generally at or near a measurement floor) across low glare intensities but rose drastically between two adjacent glare increments, suggesting that the resolution between settings selected a priori for the experiment were inadequate for proper psychophysical characterization. This trend was also observed in the subjective ratings of visual discomfort.

There was an analogous issue with the marksmanship performance assessment. The glare conditions were expected to lead to a gradual reduction in perceived target fidelity and a corresponding increase in shot group spread, but the target was effectively lost upon exposure to one of the glare intensity increments after minimal disruption. While the air rifles used in the

marksmanship assessment present a significantly lower safety hazard than firearms, they cannot be safely operated when the target is not discernable to the participant.

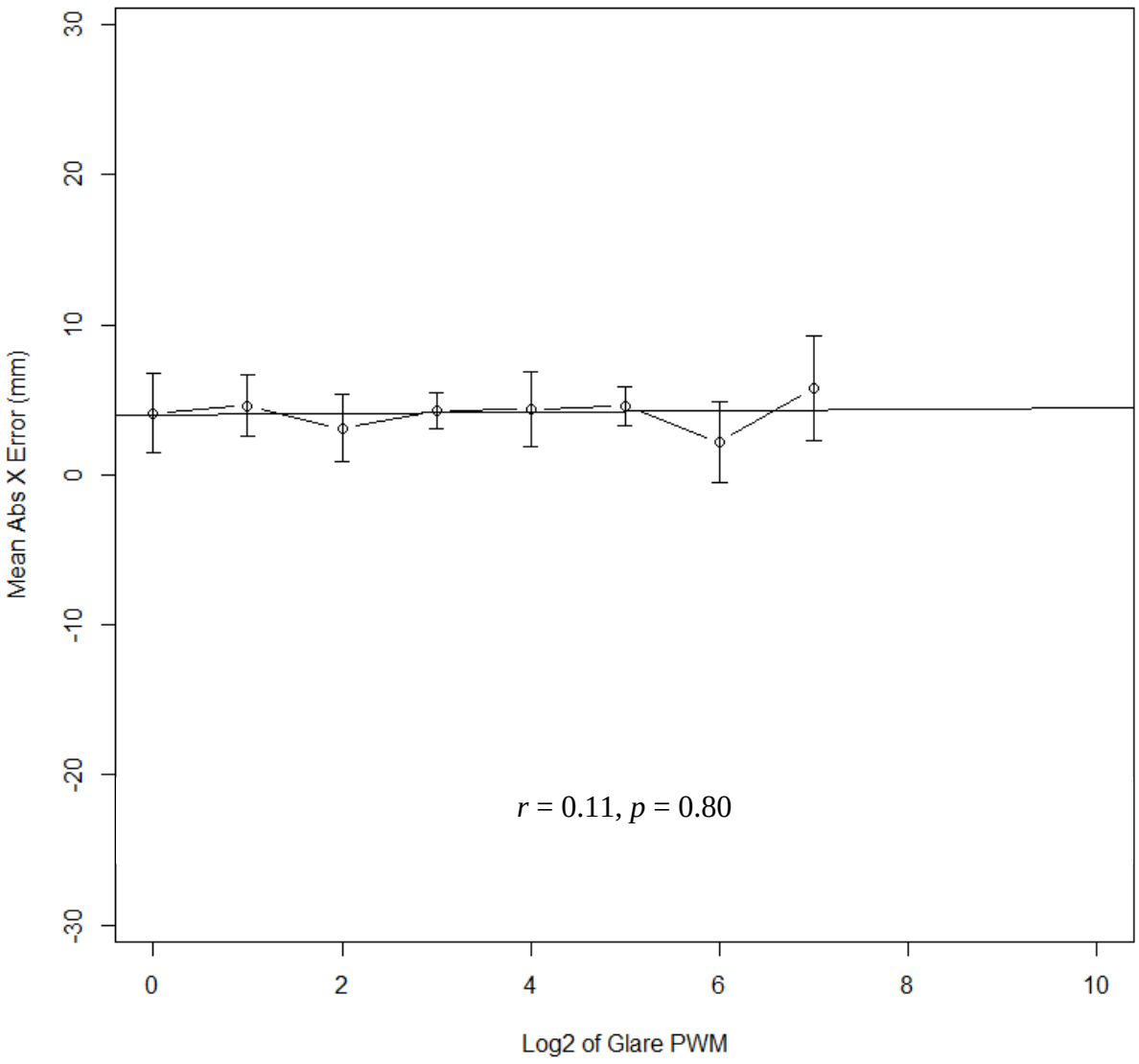

Figure 1. Uniform glare condition with absolute value of X-axis target error in millimeters (mm).

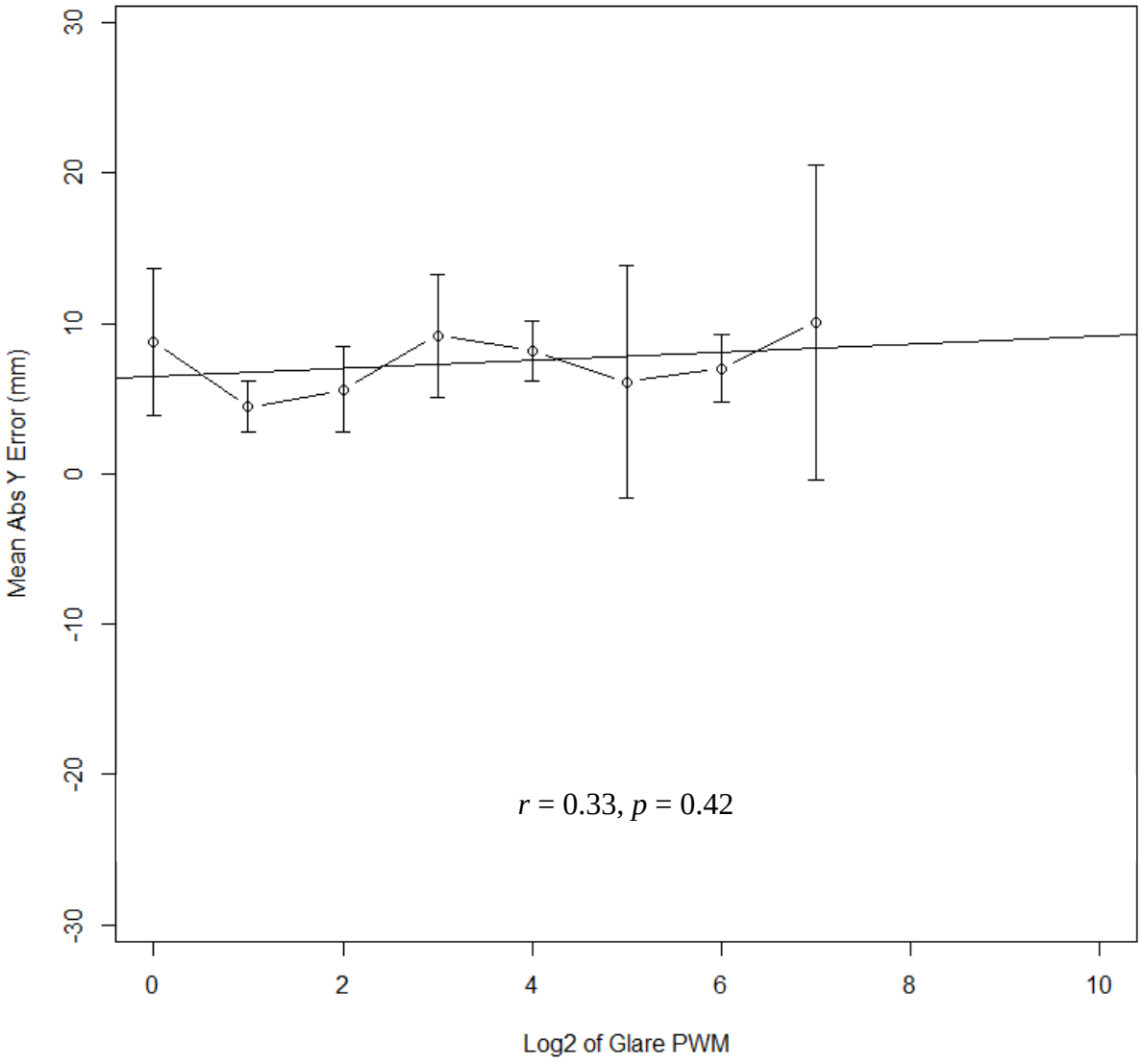

Figure 2. Uniform glare condition with absolute value of Y-axis target error in mm.

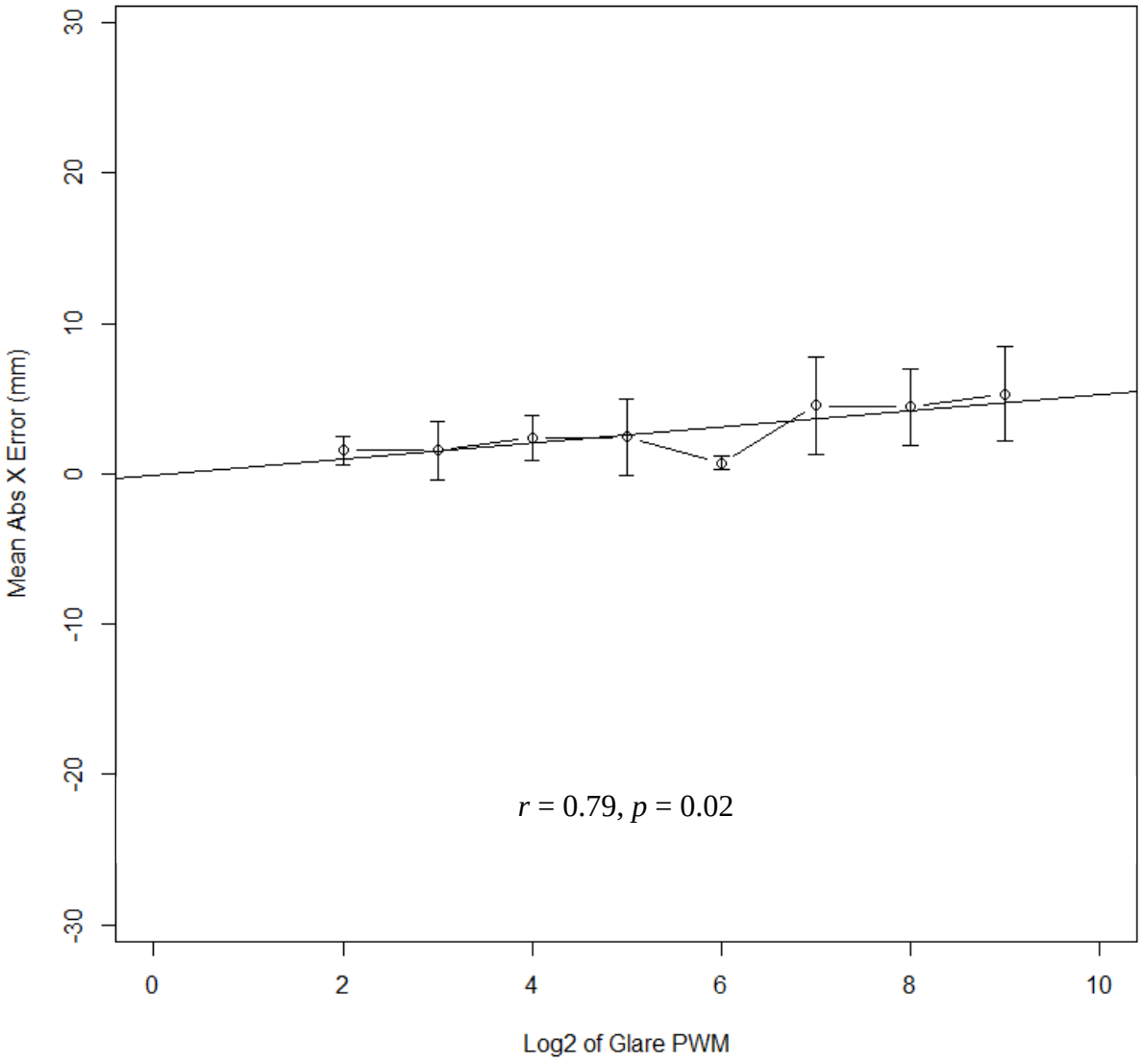

Figure 3. Top glare condition with absolute value of X-axis target error in mm.

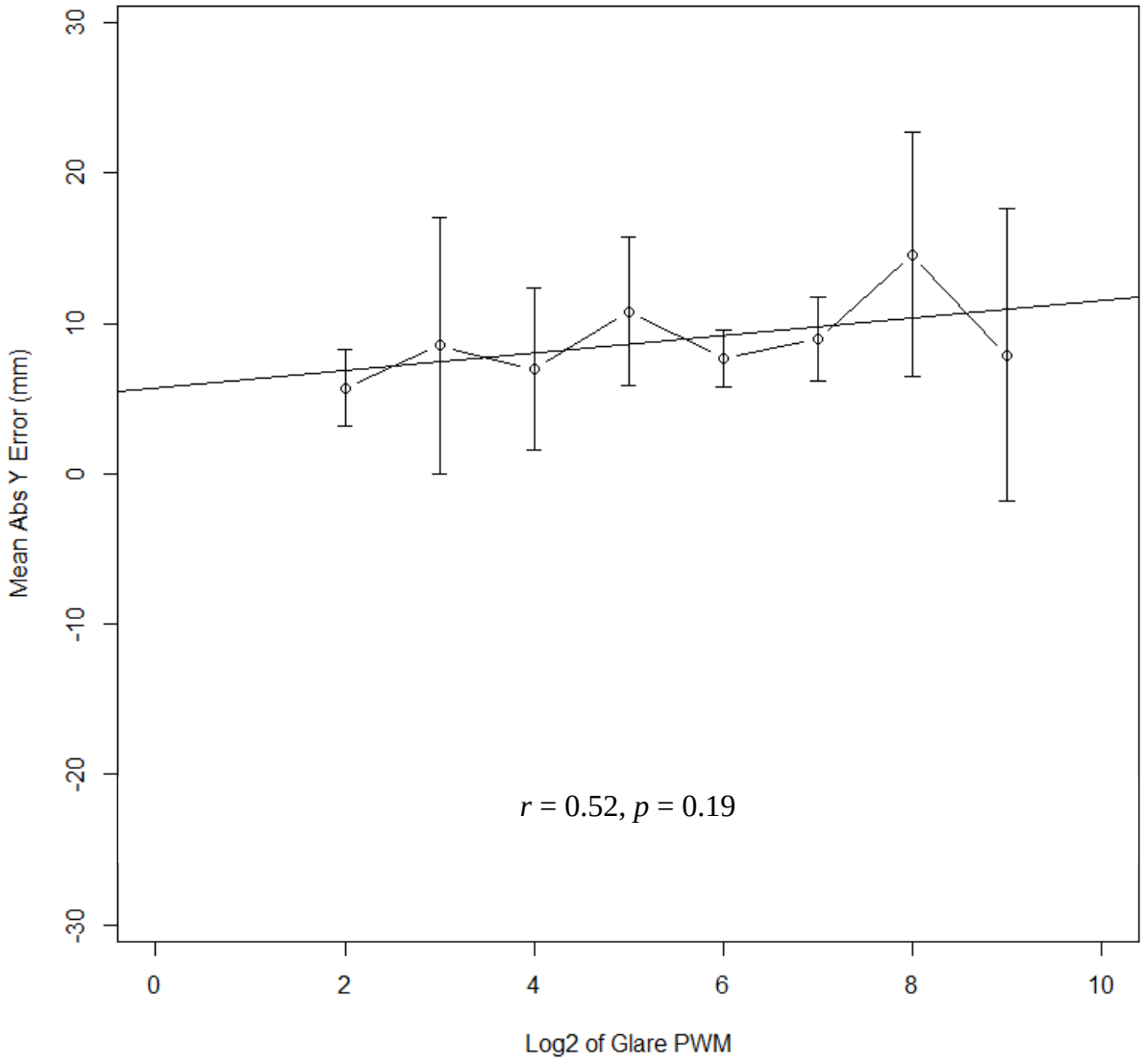

Figure 4. Top glare condition with absolute value of Y-axis target error in mm.

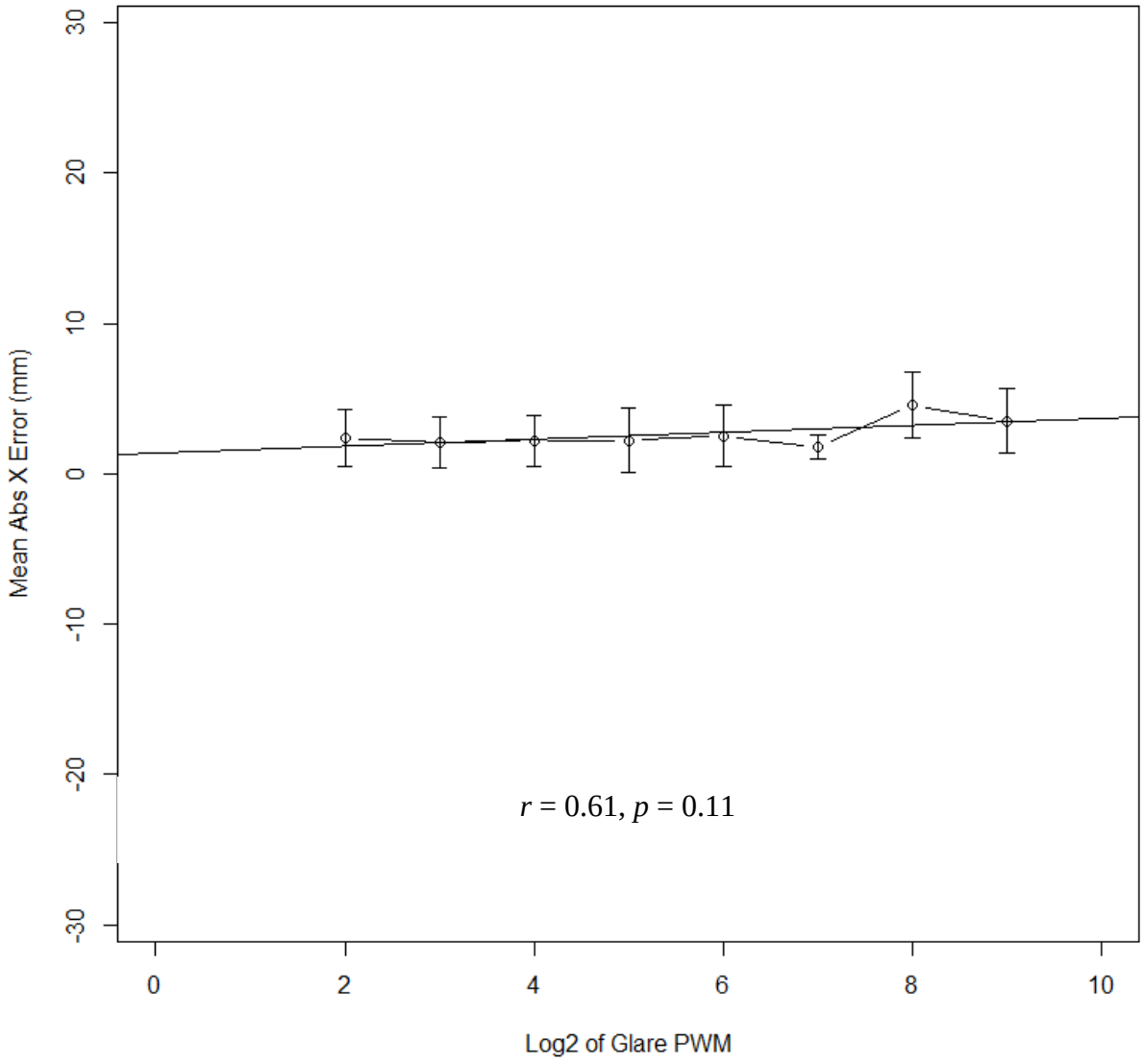

Figure 5. Bottom glare condition with absolute value of X-axis target error in mm.

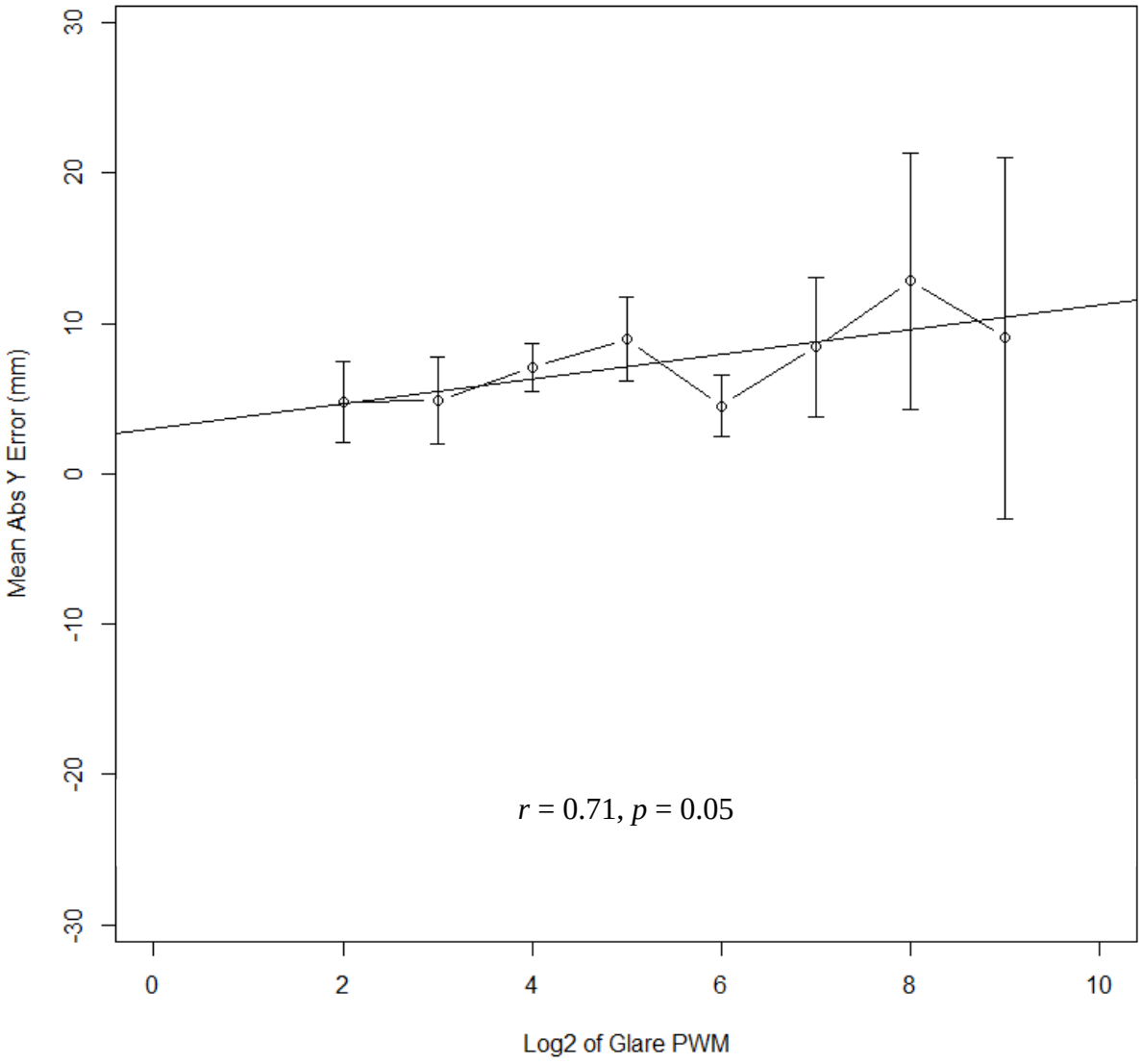

Figure 6. Bottom glare condition with absolute value of Y-axis target error in mm.

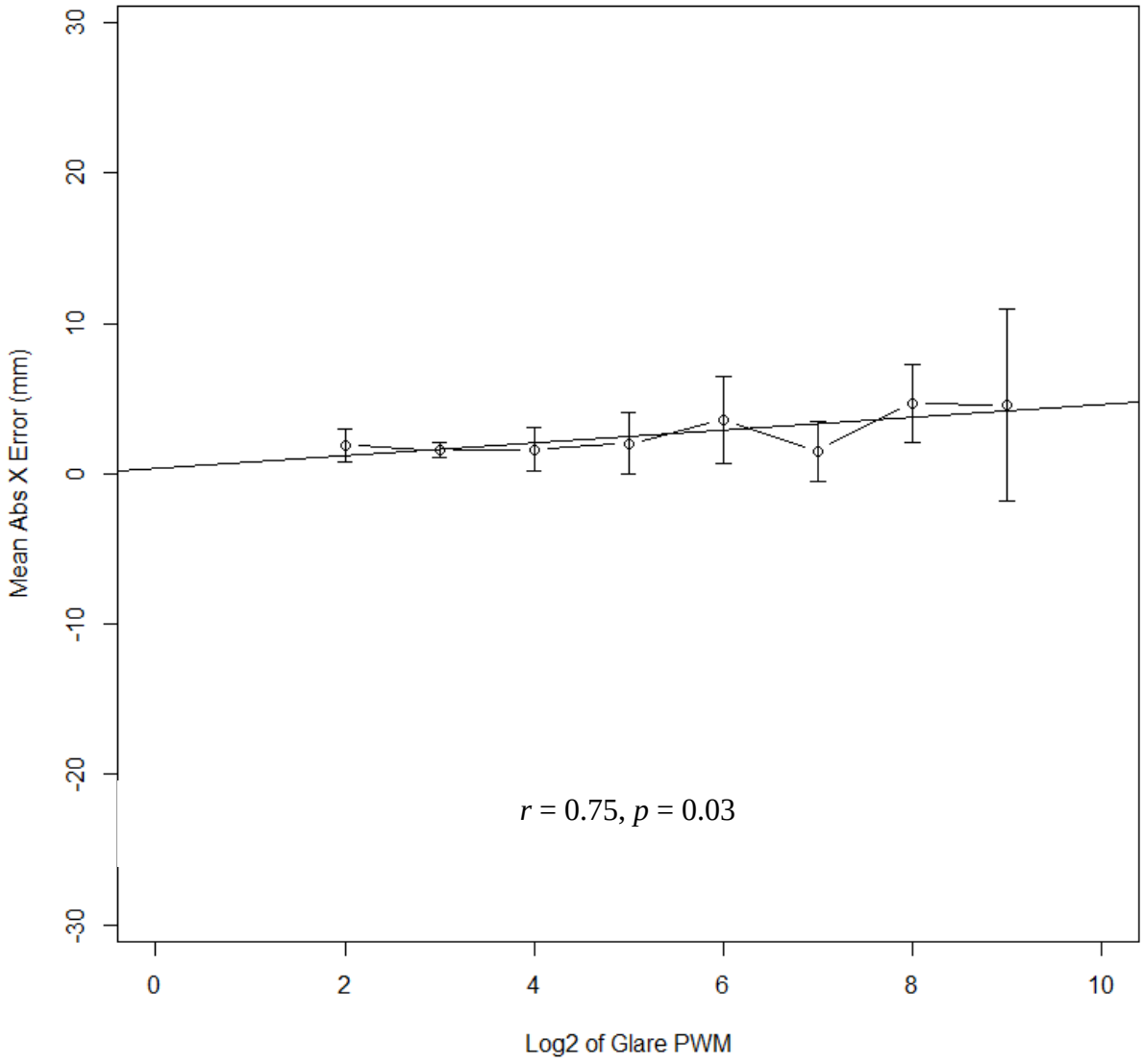

Figure 7. Left side glare condition with absolute value of X-axis target error in mm.

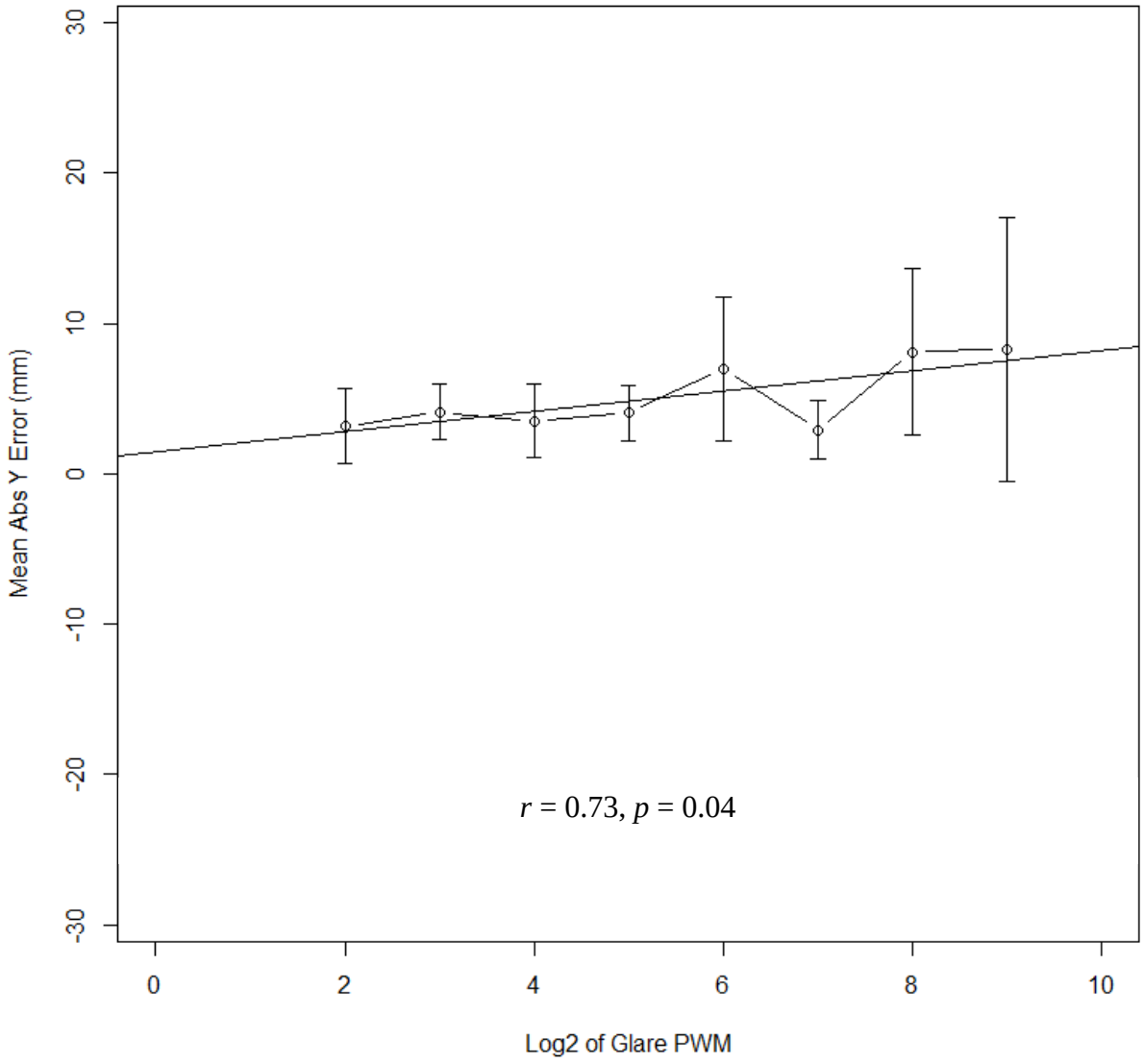

Figure 8. Left side glare condition with absolute value of Y-axis target error in mm.

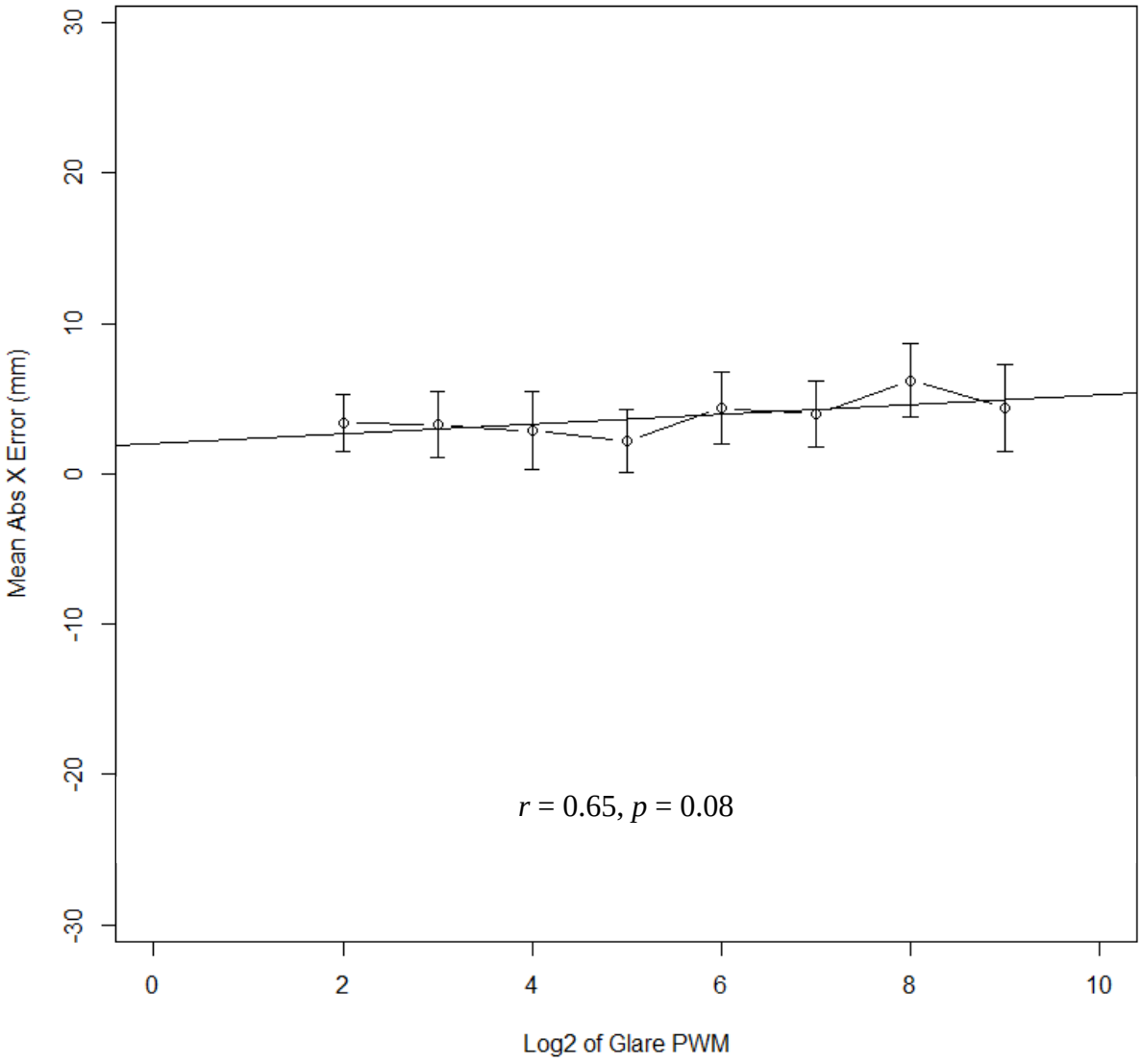

Figure 9. Right side glare condition with absolute value of X-axis target error in mm.

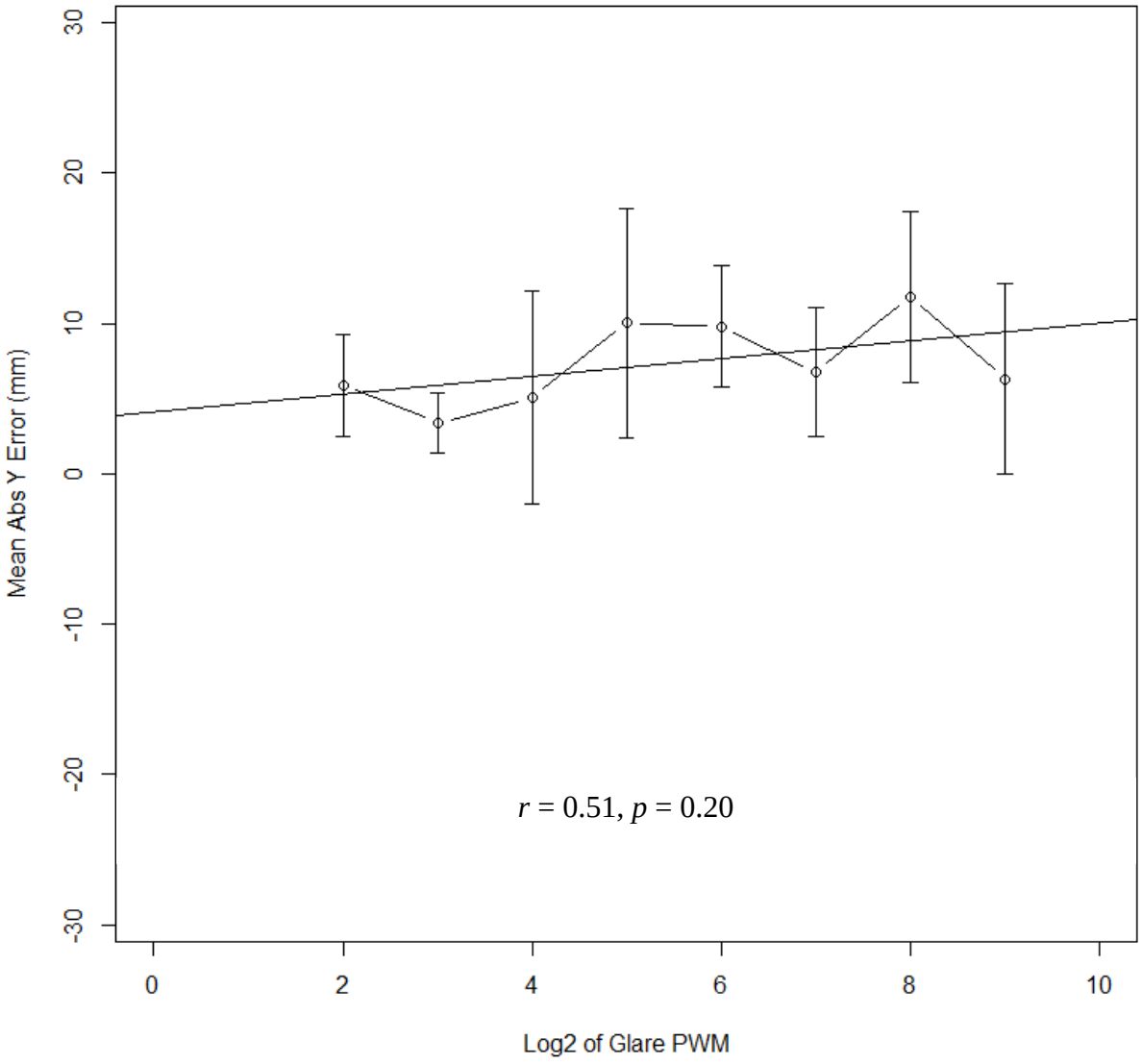

Figure 10. Right side glare condition with absolute value of Y-axis target error in mm.

In all angular orientations (as well as the uniform condition) and for both the X- and Y-axes, average error in marksmanship performance generally increased with glare intensity, but the correlations were not statistically significant if correcting for multiple comparisons.

While generally unremarkable, the visual performance measures, subjective discomfort ratings, and MPOD assessments potentially bear more similarity to clinical diagnostics than would marksmanship performance scores. In light of this, due to the extreme sample size restriction, visualizations are intentionally omitted from this document.

Discussion

The experiment was halted early for two reasons. First, the experimental design was inadequate because it was created around untested, novel equipment and the regulatory space did not permit flexible specification of stimuli conditions. Secondly, between proposal of the project and the end of data collection, USAARL's assigned research priorities were shifted as part of the Army's modernization effort.

Regardless, there has been noteworthy success associated with the project. The design of the proximal glare source has been incorporated into a pending patent (# 62/808,963) as an expanded capability. Additionally, software developed for this experiment likely has uses beyond the scope of the project, including potential clinical applications.

Recommendations

Countermeasures for glare should be incorporated into marksmanship focused roles, equipment, and practices. Prior examinations of the impact of MPOD on glare disability (Stringham & Hammond, 2008; Putnam & Bassi, 2015) and the impact of dietary supplementation on glare disability (Hammond, et al., 2014; Stringham, O'Brien, & Stringham, 2016) suggest that supplementation of lutein and zeaxanthin may be advisable for soldiers facing issues with glare. As lutein and zeaxanthin both are generally recognized as safe (U.S. Food and Drug Administration [FDA] GRNs No. 110 [2003], 140 [2004], 221 [2007], 291 [2009], 385 [2011], 390 [2012], 416 [2012], 432 [2012], 481 [2014], 542 [2015], 543 [2015], 550 [2015], 588 [2016], and 639 [2016]) and currently available data suggests no toxicity concerns (Harikumar et al., 2008; Granado et al., 2003; Nidhi & Baskaran, 2013; Thurnham & Howard, 2013; Edwards, 2016), supplementation may be advisable for all military personnel.

Conclusion

Despite this experiment not yielding the intended predictive model, the negative impacts of glare on visual performance appear to carry over into the domain of marksmanship. Additionally, if there is clinical interest in testing Vernier acuity (especially in austere settings with space constraints, as may be common in military environments), the sub-pixel rendering method developed for this experiment may significantly reduce the needed viewing distance.

References

Bailey, I. L., & Lovie, J. E. (1976). New design principles for visual acuity letter charts. *American Journal of Optometry and Physiological Optics, 53*(11), 740-745.

Banister, J. M. (1893). Rifle practice in its relations to eye-strain. *The American Journal of the Medical Sciences, 106*(5), 552.

Edwards, J. A. (2016). Zeaxanthin: Review of toxicological data and acceptable daily intake. *Journal of Ophthalmology*.

Granado, F., Olmedilla, B., & Blanco, I. (2003). Nutritional and clinical relevance of lutein in human health. *British Journal of Nutrition, 90*(3), 487-502.

Hammond, B. R., Fletcher, L. M., Roos, F., Wittwer, J., & Schalch, W. (2014). A double-blind, placebo-controlled study on the effects of lutein and zeaxanthin on photostress recovery, glare disability, and chromatic contrast. *Investigative Ophthalmology & Visual Science, 55*(12), 8583-8589.

Harikumar, K. B., Nimita, C. V., Preethi, K. C., Kuttan, R., Shankaranarayana, M. L., & Deshpande, J. (2008). Toxicity profile of lutein and lutein ester isolated from marigold flowers (Tagetes erecta). *International Journal of Toxicology, 27*(1), 1-9.

Nidhi, B., & Baskaran, V. (2013). Acute and subacute toxicity assessment of lutein in lutein-deficient mice. *Journal of Food Science, 78*(10), T1636-T1642.

Putnam, C. M., & Bassi, C. J. (2015). Macular pigment spatial distribution effects on glare disability. *Journal of Optometry, 8*(4), 258-265.

Stringham, J. M., & Hammond, B. R. (2008). Macular pigment and visual performance under glare conditions. *Optometry and Vision Science, 85*(2), 82-88.

Stringham, J. M., O'Brien, K. J., & Stringham, N. T. (2016). Macular carotenoid supplementation improves disability glare performance and dynamics of photostress recovery. *Eye and Vision, 3*(1), 1-8.

Thurnham, D. I., & Howard, A. N. (2013). Studies on meso-zeaxanthin for potential toxicity and mutagenicity. *Food and Chemical Toxicology, 59*, 455-463.

U.S. Food and Drug Administration. (2003). Lutein esters: GRN No. 110. https://www.cfsanappsexternal.fda.gov/scripts/fdcc/index.cfm?set=GRASNotices&id=110

U.S. Food and Drug Administration. (2004). Crystalline lutein: GRN No. 140. https://www.cfsanappsexternal.fda.gov/scripts/fdcc/index.cfm?set=GRASNotices&id=140

U.S. Food and Drug Administration. (2007). Suspended lutein: GRN No. 221.

https://www.cfsanappsexternal.fda.gov/scripts/fdcc/index.cfm?set=GRASNotices&id=221

U.S. Food and Drug Administration. (2009). Crystalline lutein: GRN No. 291. https://www.cfsanappsexternal.fda.gov/scripts/fdcc/index.cfm?set=GRASNotices&id=291

U.S. Food and Drug Administration. (2011). Lutein: GRN No. 385. https://www.cfsanappsexternal.fda.gov/scripts/fdcc/index.cfm?set=GRASNotices&id=385

U.S. Food and Drug Administration. (2012). Suspended lutein: GRN No. 390. https://www.cfsanappsexternal.fda.gov/scripts/fdcc/index.cfm?set=GRASNotices&id=390

U.S. Food and Drug Administration. (2012). Lutein diacetate: GRN No. 416. https://www.cfsanappsexternal.fda.gov/scripts/fdcc/index.cfm?set=GRASNotices&id=416

U.S. Food and Drug Administration. (2012). Lutein diacetate: GRN No. 432. https://www.cfsanappsexternal.fda.gov/scripts/fdcc/index.cfm?set=GRASNotices&id=432

U.S. Food and Drug Administration. (2014). Meso-zeaxanthin: GRN No. 481. https://www.cfsanappsexternal.fda.gov/scripts/fdcc/index.cfm?set=GRASNotices&id=481

U.S. Food and Drug Administration. (2015). Lutein: GRN No. 542. https://www.cfsanappsexternal.fda.gov/scripts/fdcc/index.cfm?set=GRASNotices&id=542

U.S. Food and Drug Administration. (2015). Lutein esters: GRN No. 543. https://www.cfsanappsexternal.fda.gov/scripts/fdcc/index.cfm?set=GRASNotices&id=543

U.S. Food and Drug Administration. (2015). Meso-zeaxanthin: GRN No. 550. https://www.cfsanappsexternal.fda.gov/scripts/fdcc/index.cfm?set=GRASNotices&id=550

U.S. Food and Drug Administration. (2015). Zeaxanthin from Capsicum annum (paprika): GRN No. 588. https://www.cfsanappsexternal.fda.gov/scripts/fdcc/index.cfm?set=GRASNotices&id=588

U.S. Food and Drug Administration. (2016). Zeaxanthin from marigold: GRN No. 639. https://www.cfsanappsexternal.fda.gov/scripts/fdcc/index.cfm?set=GRASNotices&id=639

Appendix A. Custom Python Code

```
#Tested in Python 3.6.4

windowTitle = "sub_pixel_vernier_06"



from tkinter import *

from tkinter.ttk import *

from tkinter import filedialog

from tkinter import messagebox

import random

import time #time.strftime("%Y-%m-%d--%H-%M-%S") #This spits out Year-Month-Day--Hour(24)-Minute-Second



import pprint

import math



import argparse

parser = argparse.ArgumentParser()

parser.add_argument("--barColor", type = int, nargs = 1, help = "Select value 0-255. Background will be complementary.", default = [0])

parser.add_argument("--displayWidth", type = int, nargs = 1, help = "Display width in pixels.", default = [1200])

parser.add_argument("--displayHeight", type = int, nargs = 1, help = "Display height in pixels.", default = [800])

parser.add_argument("--barWidth", type = int, nargs = 1, help = "Bar width in pixels.", default = [50])

parser.add_argument("--barHeight", type = int, nargs = 1, help = "Bar height in pixels.", default = [300])

parser.add_argument("--barSeparation", type = int, nargs = 1, help = "Gap distance, in
```

```
pixels, between bars", default = [30])

        parser.add_argument("--startingOffset", type = int, nargs = 1, help = "Lateral offset, in sub-pixels, between bars during initial presentation.", default = [900])

        parser.add_argument("--step", type = float, nargs = 1, help = "Proportion (0 to 1) decrease in lateral offset with a correct response. Incorrect responses result in an increase in gap width equal to multiplying by the inverse.", default = [0.8])

        parser.add_argument("--numErrors", type = int, nargs=1, help = "Number of times errors are permitted before exiting the testing routine.", default = [5])

        parser.add_argument("--blankDuration", type=float, nargs=1, help = "(Float) Duration in seconds for blank screen between updates to bar locations.", default = [0.50])

        args = parser.parse_args()



        class SubPixelBar:

                def __init__(self, canvas = None, color=0, width=10, height=50, top_corner_x=100, top_corner_y=100):

                        self.canvas = canvas

                        if((color > 255) or (color < 0)): #Check that color is appropriate within 8-bit range

                                print("ERROR - INVALID COLOR")

                                root.quit()

                        else:

                                self.color_val = color

                                self.color_comp = abs(255-self.color_val)

                                self.color = [self.color_val,self.color_val,self.color_val]

                                self.bg_color = [self.color_comp,self.color_comp,self.color_comp]

                                #The order of these will need to be changed if the arrangement of subpixels varies, but these work with R|G|B ordered subpixels.

                                self.sub_colors_negative = [[self.color_comp,self.color_comp,self.color_comp],[self.color_comp,self.color_val,self.color_val],[self.color_comp,self.color_comp,self.color_val]]
```

```
                self.sub_colors_positive =
[[self.color_val,self.color_val,self.color_val],[self.color_val,self.color_comp,self.color_comp],[s
elf.color_val,self.color_val,self.color_comp]]

                self.width = width-1 #The '-1' accounts for the sub-pixel rendering
component

                self.height = height

                self.top_corner_x = top_corner_x

                self.top_corner_y = top_corner_y

                self.sub_x_offset = 0

                #Center component of bar

                self.bar_starting_coords = [self.top_corner_x, self.top_corner_y,
self.top_corner_x+self.width, self.top_corner_y+self.height]

                self.bar = self.canvas.create_rectangle(self.bar_starting_coords, fill =
self.generateColorString(self.color), width=0)

                #Left sub-pixel rendered component of bar

                self.sub_bar_left_starting_coords = [self.top_corner_x-1,
self.top_corner_y, self.top_corner_x-1, self.top_corner_y+self.height]

                self.sub_bar_left =
self.canvas.create_line(self.sub_bar_left_starting_coords,
fill=self.generateColorString(self.sub_colors_negative[0]), width=1)

                #Right sub-pixel rendered component of bar

                self.sub_bar_right_starting_coords = [self.top_corner_x+self.width,
self.top_corner_y, self.top_corner_x+self.width, self.top_corner_y+self.height]

                self.sub_bar_right =
self.canvas.create_line(self.sub_bar_right_starting_coords, fill=
self.generateColorString(self.sub_colors_positive[0]), width=1)



        def generateColorString(self, colorArray):

                out_string =
"#"+str(hex(colorArray[0])[2:].zfill(2))+str(hex(colorArray[1])[2:].zfill(2))+str(hex(colorArray[2
])[2:].zfill(2)) #Tkinter requires colors be specified as hex strings
```

```
                return out_string


        def offset(self, off_x = 0, off_y = 0):

                self.canvas.coords(self.bar,
[self.bar_starting_coords[0]+off_x,self.bar_starting_coords[1]+off_y,self.bar_starting_coords[2]
+off_x,self.bar_starting_coords[3]+off_y])

                self.canvas.coords(self.sub_bar_left,
[self.sub_bar_left_starting_coords[0]+off_x,self.sub_bar_left_starting_coords[1]+off_y,self.sub_
bar_left_starting_coords[2]+off_x,self.sub_bar_left_starting_coords[3]+off_y])

                self.canvas.coords(self.sub_bar_right,
[self.sub_bar_right_starting_coords[0]+off_x,self.sub_bar_right_starting_coords[1]+off_y,self.s
ub_bar_right_starting_coords[2]+off_x,self.sub_bar_right_starting_coords[3]+off_y])



        def set_sub_x_offset(self, sub_offset=0):

                if(sub_offset == 0):#Under this condition, the left bar should be the color
of the background and the right bar should be the color of the target

                        self.canvas.itemconfig(self.sub_bar_left, fill =
self.generateColorString(self.sub_colors_negative[0]))

                        self.canvas.itemconfig(self.sub_bar_right, fill =
self.generateColorString(self.sub_colors_positive[0]))

                elif(sub_offset < 0):

                        if(sub_offset < -2): #If one or more full pixels of adjustment are
available.

                                self.offset((sub_offset//3),0)

                                self.set_sub_x_offset(sub_offset%3) #Recursive

                        else:

                                self.canvas.itemconfig(self.sub_bar_left, fill =
self.generateColorString(self.sub_colors_negative[sub_offset]))

                                self.canvas.itemconfig(self.sub_bar_right, fill =
self.generateColorString(self.sub_colors_positive[sub_offset]))

                elif(sub_offset > 0):
```

```
                    if(sub_offset > 2): #If one or more full pixels of adjustment are
available.
                        self.offset((sub_offset//3),0)
                        self.set_sub_x_offset(sub_offset%3) #Recursive
                    else:
                        self.canvas.itemconfig(self.sub_bar_left, fill =
self.generateColorString(self.sub_colors_negative[sub_offset]))
                        self.canvas.itemconfig(self.sub_bar_right, fill =
self.generateColorString(self.sub_colors_positive[sub_offset]))


        #The hide and reveal methods require updating the application, so the update time
should be incorporated into delays in anything that's timing sensitive
        def hide(self):
            self.canvas.itemconfig(self.bar, state="hidden")
            self.canvas.itemconfig(self.sub_bar_right, state="hidden")
            self.canvas.itemconfig(self.sub_bar_left, state="hidden")
            root.update()


        def reveal(self):
            self.canvas.itemconfig(self.bar, state="normal")
            self.canvas.itemconfig(self.sub_bar_right, state="normal")
            self.canvas.itemconfig(self.sub_bar_left, state="normal")
            root.update()


    #Break out args into more manageable variables
    barColor = args.barColor[0]
    backGroundColor = abs(255-barColor) #This approach allows for adjustable contrast
```

over the full 8-bit brightness range with subpixel rendering at intermediate brightnesses

```
	backGroundColorString = "#"+str(hex(backGroundColor)[2:].zfill(2))+str(hex(backGroundColor)[2:].zfill(2))+str(hex(backGroundColor)[2:].zfill(2))

	displayWidth = args.displayWidth[0]

	displayHeight = args.displayHeight[0]

	barWidth = args.barWidth[0]

	barHeight = args.barHeight[0]

	barSeparation = args.barSeparation[0]

	startingOffset = args.startingOffset[0]

	step = args.step[0]

	numErrors = args.numErrors[0]

	blankDuration = args.blankDuration[0]


	global saveFile, errorResponsePixelOffsets, trialNumber, cumulativeNumErrors

	errorResponsePixelOffsets = []

	trialNumber = 1

	cumulativeNumErrors = 0


	root = Tk()

	root.title(windowTitle)


	display = Canvas(root, width = displayWidth, height = displayHeight, bg = backGroundColorString)

	display.pack()
```

```
    center_x = displayWidth//2

    center_y = displayHeight//2


    topBar = SubPixelBar(canvas = display, color = barColor, width=barWidth,
height=barHeight, top_corner_x = (center_x-(barWidth//2)), top_corner_y = (center_y-
(barHeight)-(barSeparation//2)))

    bottomBar = SubPixelBar(canvas = display, color = barColor, width=barWidth,
height=barHeight, top_corner_x = (center_x-(barWidth//2)), top_corner_y =
(center_y+(barSeparation//2)))


    def writeHeader(saveFile):

        saveFile.write("Trial_Number,TimeStamp,Sub_Pixel_Offset,Subject_Decision,Er
ror_Message\n")


    def writeResponse(saveFile, trialNumber, sub_pixel_offset, subject_decision):

        saveFile.write("{},{},{},{},{}\n".format(trialNumber,time.strftime("%H:%M:%S
"),sub_pixel_offset,subject_decision,")) #If time stamps need date stamps as well, a global find
and replace of the strftime string should be safe


    def writeError(saveFile, error):

        saveFile.write("NA,{},NA,NA,{}".format(time.strftime("%H:%M:%S"), error))


    def writeZeroExit(saveFile):

        writeError(saveFile, "writeZeroExitError")

        root.quit()


    def writeNumIncorrectExit(saveFile):

        writeError(saveFile, "writeNumIncorrectExitError")
```

```
            root.quit()


        def updateStimulus():

            global topBar, bottomBar, trialNumber, currentOffset, lastSelected, step,
cumulativeNumErrors, numErrors, saveFile, blankDuration, errorResponseSubpixelOffsets

            if(saveFile is None):

                messagebox.showwarning("Error","No save file selected. Exiting abrupty
and ungracefully.")

                root.quit()

            if((currentOffset > 0) and (lastSelected == "Right")) or ((currentOffset < 0) and
(lastSelected == "Left")):

                #print("Correct")

                writeResponse(saveFile, trialNumber, currentOffset, "Correct")

                if(abs(currentOffset)<7): #Go to individual subpixel adjustments if 2 or
less pixels of difference. This may need to change.

                    topBar.hide()

                    bottomBar.hide()

                    time.sleep(blankDuration)

                    currentOffset = int(random.choice([-1,1])*(abs(currentOffset)-1))

                    bottomBar.set_sub_x_offset(currentOffset)


                else:

                    topBar.hide()

                    bottomBar.hide()

                    time.sleep(blankDuration)

                    currentOffset = int(random.choice([-1,1])*step*currentOffset)

                    bottomBar.set_sub_x_offset(currentOffset)
```

```
        else:
            #print("Incorrect")
            writeResponse(saveFile, trialNumber, currentOffset, "Incorrect")
            if(abs(currentOffset)<7): #Go to individual subpixel adjustments if 2 or less pixels of difference. This may need to change.
                topBar.hide()
                bottomBar.hide()
                time.sleep(blankDuration)
                currentOffset = int(random.choice([-1,1])*(abs(currentOffset)+1))
                bottomBar.set_sub_x_offset(currentOffset)

            else:
                topBar.hide()
                bottomBar.hide()
                time.sleep(blankDuration)
                currentOffset = int(random.choice([-1,1])*(1/step)*currentOffset)
                bottomBar.set_sub_x_offset(currentOffset)

            cumulativeNumErrors += 1
        if(currentOffset == 0):
            writeZeroExit(saveFile)
        elif(cumulativeNumErrors >= numErrors):
            writeNumIncorrectExit(saveFile)
        else:
            topBar.reveal()
```

```
                bottomBar.reveal()
            trialNumber += 1


        root.update() #Without this line, the keyboard binding breaks


        global saveFile, currentOffset, errorResponseSubpixelOffsets
        errorResponseSubpixelOffsets = []


        saveFile = filedialog.asksaveasfile(mode='a', initialfile = "*.csv", filetypes = (("CSV (MS-DOS)","*.csv"),("all files","*.*")))
        if(saveFile is not None):
            writeHeader(saveFile)


        currentOffset = random.choice([-1,1])*startingOffset
        bottomBar.set_sub_x_offset(currentOffset)


        def leftBinding(event):
            global lastSelected
            lastSelected = "Left"
            updateStimulus()


        def rightBinding(event):
            global lastSelected
            lastSelected = "Right"
            updateStimulus()
```

```
root.bind("<Right>", leftBinding)
root.bind("<Left>", rightBinding)



def escapeBinding(event):
    writeError(saveFile, "Exit initiated via escape key")
    root.quit()
root.bind("<Escape>", escapeBinding)


root.mainloop()
saveFile.close()#This will not properly close the CSV file if the program is shut down from the terminal, but it does handle the escape key an
```

U.S. Army Aeromedical Research Laboratory
Fort Rucker, Alabama

All of USAARL's science and technical information documents are available for download from the Defense Technical Information Center.

https://discover.dtic.mil/results/?q=USAARL

Army Futures Command
U.S. Army Medical Research and Development Command