Submission Template for ACM Papers

Visualizing references to off-screen content on mobile devices: A comparison of Wedges and Edge Hashes

Ryan, A, Pybus

University of British Columbia, 3333 University Way, Kelowna, BC, ryanpybus8596@gmail.com

ABSTRACT

To combat the limitations of small screen devices, researchers have created techniques to indicate the locations of off-screen targets. Techniques such as Wedge and Halo have been used to convey direction and distance but occupy a significant amount of onscreen space as they use size to convey distance. We investigate Edge Hashes, a proposed technique to minimize space used. We compare its effectiveness to the established wedge, looking at how density and range affect the comparison, testing users' ability to determine the nearest and furthest targets from a group. We find Wedge to be faster and more accurate than Edge Hashes, with no effect of range, and with a significant effect of density, with high density arrangements further favoring Wedge.

CCS CONCEPTS • Visualization • Off-Screen Selection • Maps • Small Screens

GitHub repo: https://github.com/RyanPybus/341Project/tree/master

Video: https://www.youtube.com/watch?v=roUdS6g8d_w

1 INTRODUCTION

Limited screen sizes of modern mobile devices pose challenges for communicating large amounts of spatial information, such as maps, large images, or web pages. The most common way to handle this is by requiring panning and zooming from the user. However, this is costly to efficiency, as it requires many actions to find a new target, namely zooming out, then panning to center a new target, before zooming back in. Another method is to represent off screen targets by on screen effects. One such method is halo [3]. Another method, which has been shown to be generally superior to halo is wedge [1]. These methods draw shapes on the screen near the edge which give information on the distance to and general direction of a given target. Shapes that can be selected by the user to snap to the target they represent. These have some limitations, however. They occupy a significant amount of screen space. They also give selection preference to some targets over others. Fitt's Law states that a target's selection time is inversely dependent on its size, and as halo and wedge use size to communicate distances, they give preference to more distant targets [5].

In this paper a new method of visualizing off screen targets is proposed and compared to a standard wedge implementation. This technique, hereafter called edge hashes, sends a ray from the target to the center of the screen. Where this ray intersects the edge, a hash is placed, with a uniformly sized icon next to it. Distance is communicated to the user by adding two other hashes, resultant from rays projected by some angle θ to each side of the central ray. The further the target is from the screen the greater the resultant spread will be.

The key findings of this paper are that edge hashes provide a slower and less accurate method for selecting the nearest and furthest off-screen targets, but occupy less space, and are less biased for selection.

- A more space efficient technique for off screen target selection
- Edge Hashes remove target selection bias
- Slower and less accurate than Wedges

2 RELATED WORK

The Edge Hash visualization is derivative of other general methods of off-screen data visualization, such as overview, halo, and particularly wedge.

Overview

Overview visualizations present a low-detail miniaturized representation of the whole alongside a higher-detail larger view of only a part of the whole. They can be presented as separate windows beside one another or overlapped. Users may scroll omnidirectionally on the detailed view or drag around a representation of the detail window located on the low-detail view [4]. The reverse of this, where the large window is the low detail view, and the user selects a location to see a smaller high detail view can be seen in magnifying popups in many applications [4]. This technique requires additional space to accommodate multiple views, and features display redundancy as the same data will be displayed twice in two levels oof detail. Additionally, the user must maintain an understanding of two different scales at the same time.

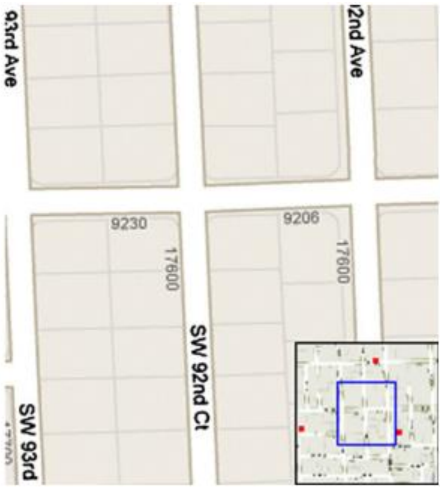


Figure 1: An example of overview on a map. [4]

Halo

Halo conveys a scale independent sense of distance by centering a circle around the target that overlaps with the edge of the screen. Users can look at these partial circles and conclude where their centers must be [3]. Halo performs well with precise distances and allows the entire display to be used for high detail viewing. Individual halos can be selected by the user to snap to their target's location. Halo techniques suffer when the number of targets in a direction becomes numerous, and displayed arcs begin to overlap. Oval arcs have been used to try to mitigate this effect [3].

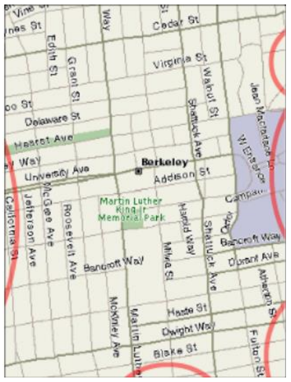


Figure 2: an example of Halo on a map. [1]

Wedge

Wedge techniques operate similarly to halo in that they use a simple shape, an acute isosceles triangle, partially displayed on screen to convey a target's location. The target is placed at the tip of the triangle, with its base and a segment of each arm visible. The target is then at the imagined intersection of these visible arms. Distance to the target is conveyed by the length of the arms [1]. This approach allows for minimizing clutter and overlap as the length of the base can be changed, as well as the rotation of the triangle.

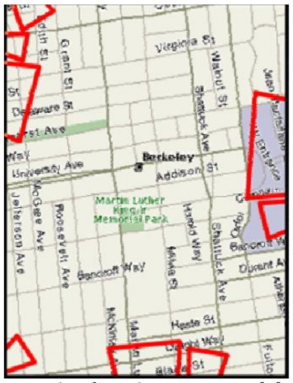


Figure 3: an example of Wedge on a map. [1]

Edge Hashes

The Wedge is the closest ancestor of the edge hash technique. The main issues we look to resolve are the wedges targeting bias, where some shapes become much larger than others, and the amount of on-screen space taken up. In real applications, targets will be both on and off screen, and any space taken up by visualizations for off screen targets will limit or clutter space for on screen targets, background information (such as road lines on a map), or for tooltips or other popups for currently selected on screen targets.

Edge hashes are displayed along the screen's edge and do not intrude far in. Selection icons are of a uniform size and provide no selection bias. The widest part of the Edge Hash visualization is at the edge of the screen rather than the middle (Figure 4). Hashes themselves provide a similar detection technique to wedges as the target is located where the three lines converge, compared to a wedge's two arms. Since the lines will be shorter, a third is added to help the user intuit the location. The Hashes are generated by drawing a line from the target to the center of the screen and placing the central hash along this line at the screen's boundary. The line is then rotated 3.5 degrees in each direction, and side hash is placed at each of these locations. Colors are generated to help differentiate nearby hashes.

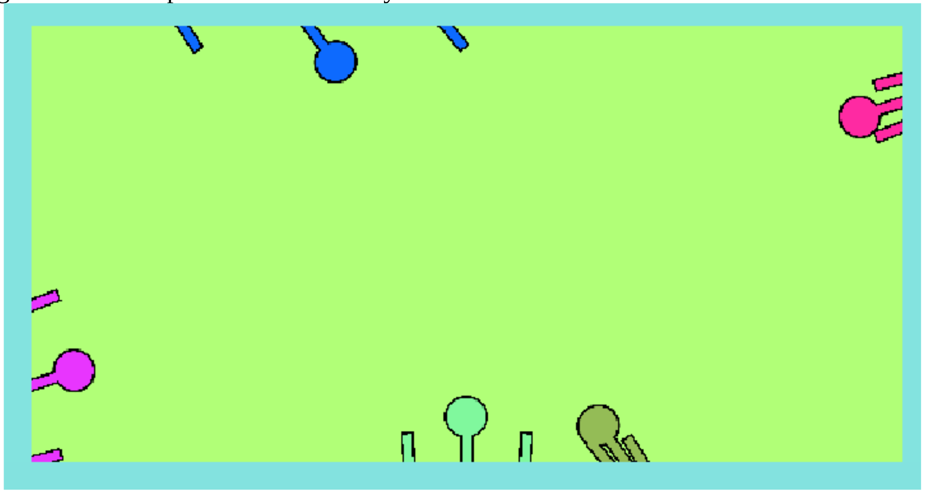


Figure 4: (Rotated 90 degrees) A screenshot of the 'screen' used in testing. Edge hashes populate the edges.

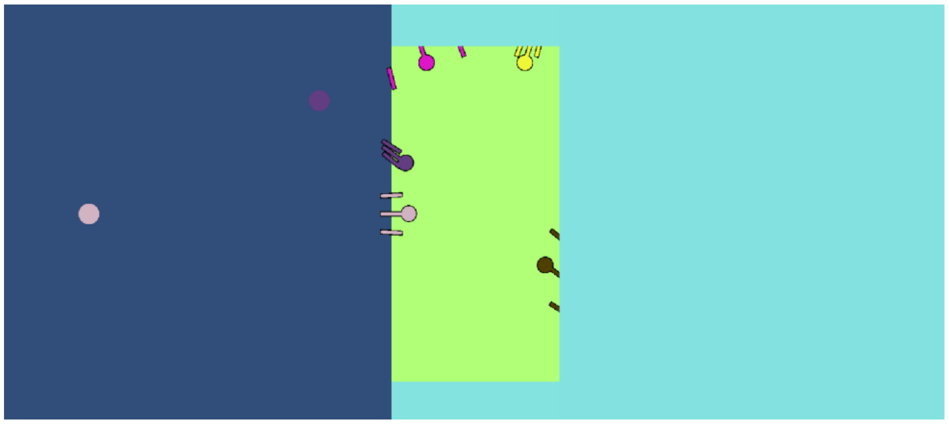


Figure 5: A screenshot of the test with the left half of the mask removed. This clearly demonstrates how the hashes are formed, and how the difference in the distance between the hashes corresponds to distance from the screen.

3 USER STUDY

The objective of this paper is to compare the effectiveness of the Edge Hash with the standard Wedge. We hypothesized that the Edge Hash will be similar in performance to the Wedge while occupying less on-screen space.

Participants

6 participants, 2 male and 4 female, between the ages of 16 and 56 participated in the study. All had normal vision, and one was familiar with both wedge and Edge Hashes.

Apparatus

The experiment operated using a custom program written in Unity that simulated a phone shaped display. This display was centered in a 16-inch 1920x1080 monitor. Selection was done using a wireless mouse.

Tasks

Furthest and Nearest. The users clicked the on-screen representation of the nearest off-screen target, then clicked the next button in the center of the screen to get the next set of targets. The next option was not available until the correct selection was made, and the number of erroneous selections was counted. This was repeated ten times each with sparse and dense target layouts. A sparse layout had five total targets off-screen, while the dense layout had ten. Targets were randomly distributed around the screen with a specified range between the nearest and furthest. A minimum rotational separation of 360/2N degrees was enforced, where N is the number of targets present. Likewise, there was a minimum difference in distance from the center of R/2N, where R is the maximum distance between the nearest and the furthest target. This was repeated with the goal changed to the furthest target from the center. This was done once with Edge Hashes and once with Wedges. The whole process took about 12 minutes.

Design

This was a within subjects experiment as access to subjects was limited. Each participant was shown an example of each visualization before the testing began. Half of the participants did Wedges then Hashes; the rest did the inverse. The dependent variables recorded were Errors (number of incorrect selections for each correct selection) made and Time (amount of time between the selection of the next button and the selection of the correct target). The independent variables were visualization type, Edge Hashes or Wedges, Density, with two levels, sparse and dense, and Range, with two levels near and far. Each participant was also asked which method they preferred. Figure 6 shows the testing screen, and Figure 7 shows some examples of densities and ranges.

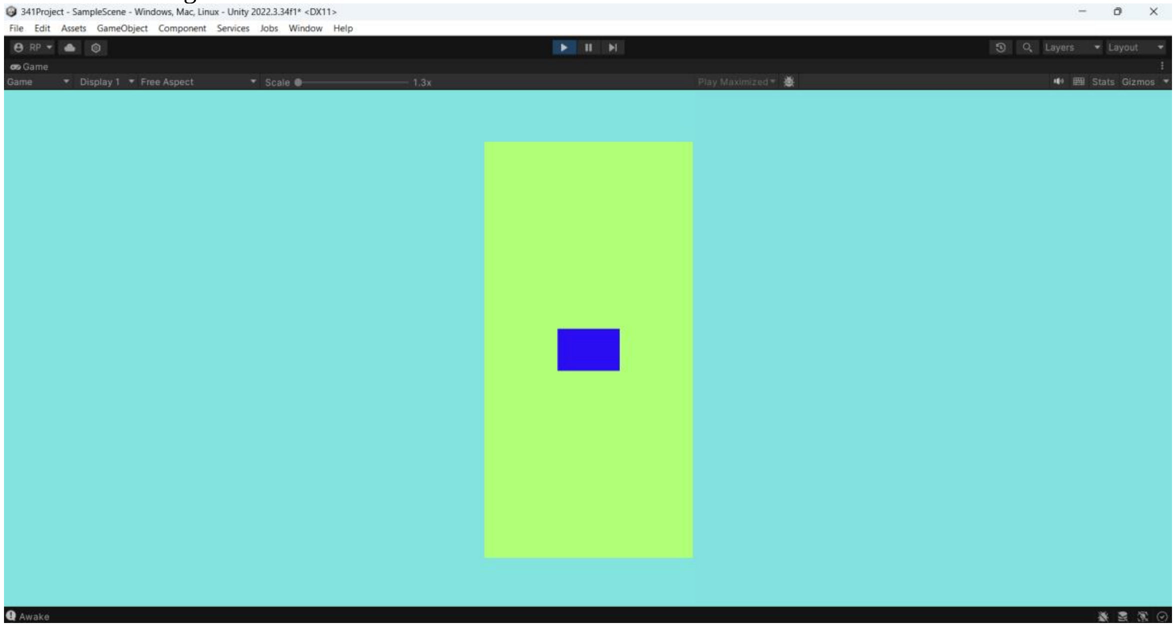
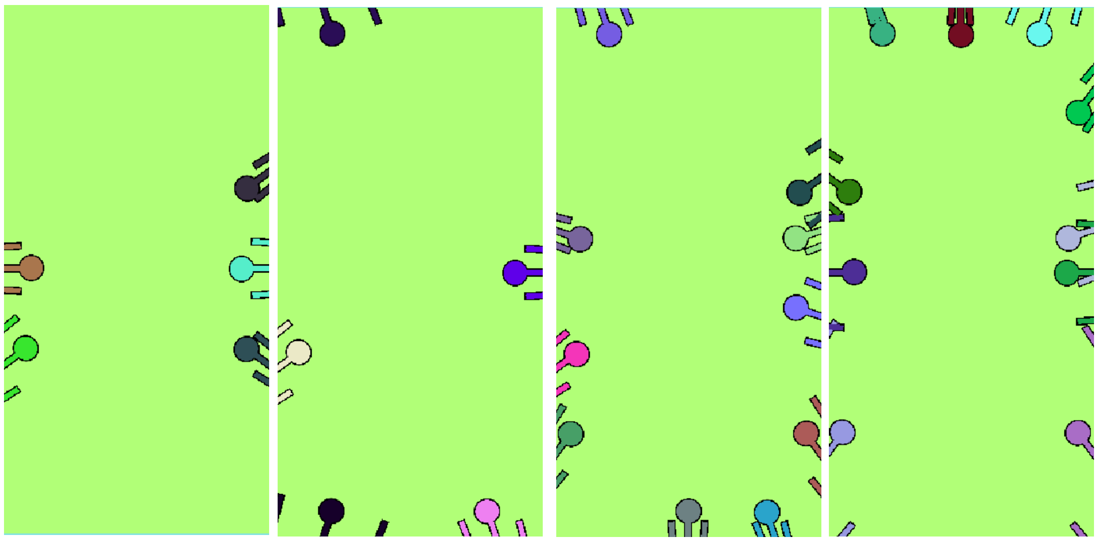


Figure 6: The entire testing screen, including the 'next' box that is to be clicked to start the time for each trial. The box is blue for the 'nearest' task, and black for the 'furthest' task.



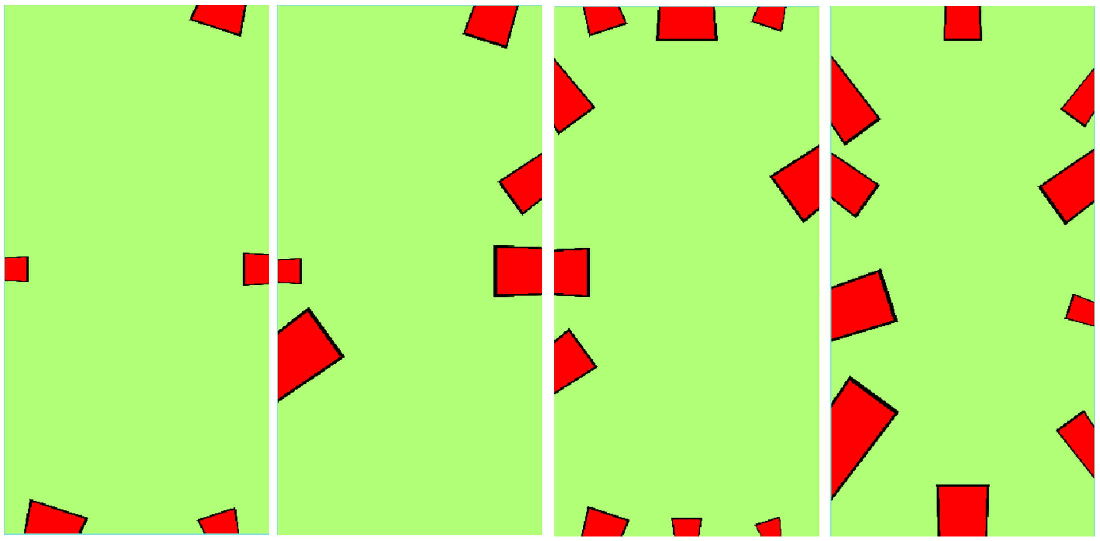


Figure 7: Testing screenshots. Top Row: Hashes (left to right) Near and sparse, far and sparse, near and dense, far and dense. Bottom Row: Wedges (left to right) Near and sparse, far and sparse, near and dense, far and dense. Blue masks have been cut out for space.

4 RESULTS

Using a two way ANOVA test with interaction, we found a statistically significant difference in average time taken by visualization type (F(1) = 341.7, p = 0), and by density(F(1) = 198.4, p = 0). We found no significant effect of range. The interaction was found to be significant (F(1) = 64.1, p = 0). A significant difference in average errors made was also found by visualization (F(1) = 57.91, p = 0), and by density (F(1) = 58.92, p = 0). No significant effect was found for range. The interaction was found to be significant (F(1) = 7.725, p < 0.01). All analysis was done in R.

Figure 8 shows the time taken for each configuration of range and density, averaged across all trials for all participants. Figure 9 shows the total errors committed by all participants for each range and density.

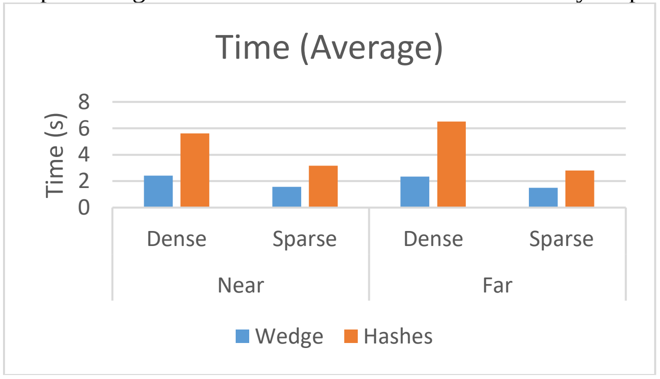


Figure 8: Average time to correctly select the target for each range, density, and visualization.

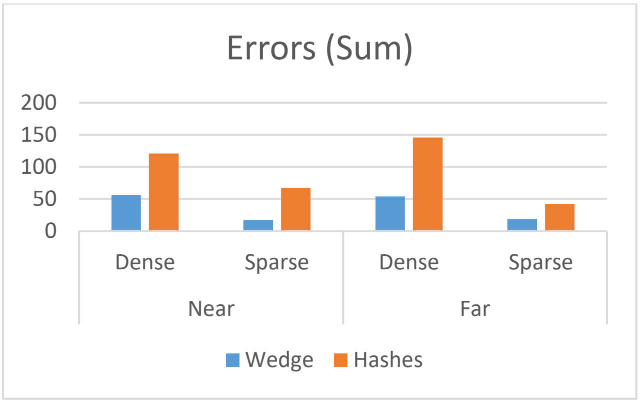


Figure 9: Total errors made by all participants at each range and density.

Of the six participants, all said that they preferred the wedge method. Common points of criticism were that Hashes became too cluttered at high densities, and that corner effects were harder to understand with Edge Hashes. Figure 3 shows the errors made by each participant.

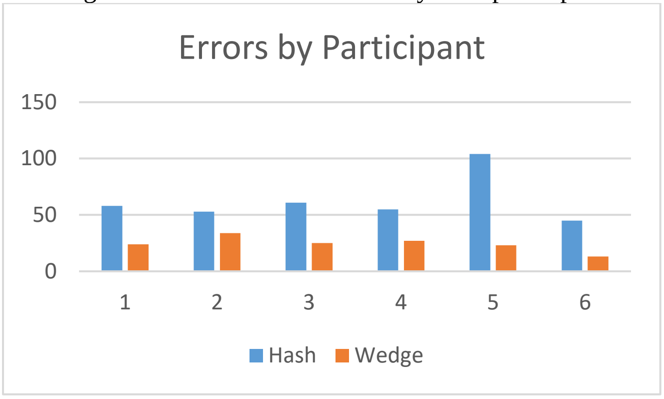


Figure 10: Total errors made by each participant. Most participants made around 60 errors in Hash, and 25 in Wedges.

5 DISCUSSION

Contrary to our hypothesis that Edge Hashes would perform similarly to Wedge, the data collected here conclusively shows that Edge Hashes is much less efficient than Wedges, as it is significantly slower and less accurate. While Range had no significant effect on speed or accuracy, Density had a deleterious effect on both. The negative effect of density however was much stronger for Edge Hashes than Wedges, meaning that for instances with many targets wedges is better by an even larger margin than with few targets. This means that the only viable use case for Edge Hashes over Wedges is when there are few targets, and you are willing to sacrifice speed and accuracy to free up a bit of screen space near the edge. An example would be something like a map browsing app, where wedges are used normally, but when an on-screen target is selected, and a popup appears, the wedges are pushed back and replaced with the more compact edge hashes.

6 CONCLUSION

We introduced a new off-screen visualization technique called Edge Hashes, which minimized the amount of screen space used to display the off-screen information. We carried out a study showing that overall, the Edge Hash method is slower and less accurate than the Wedge method. This is true for all ranges and densities tested, though Edge Hashes displays a greater negative effect on speed and accuracy for high densities. Nonetheless, Edge Hashes remains the method which occupies less screen space, and so it may be used if real estate is at a premium and the tradeoff is worth a large decrease in performance. To improve the Edge Hashes technique, we would look to reduce clutter, and to its corner performance, as because we use the screen's edge to display information, corner warping makes comparisons difficult. Further testing could look at Edge Hashes performance in tasks other than nearest and furthest.

ACKNOWLEDGMENTS

Thanks to the participants. They were not compensated.

REFERENCES

[1] Sean Gustafson, Patrick Baudisch, Carl Gutwin, and Pourang Irani. 2008. Wedge: clutter-free visualization of off-screen locations. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '08). Association for Computing Machinery, New York, NY, USA, 787–796. https://doi.org/10.1145/1357054.1357179

[2] Stefano Burigat, Luca Chittaro, and Silvia Gabrielli. 2006. Visualizing locations of off-screen objects on mobile devices: a comparative evaluation of three approaches. In Proceedings of the 8th conference on Human-computer interaction with mobile devices and services (MobileHCI '06). Association for Computing Machinery, New York, NY, USA, 239–246. https://doi.org/10.1145/1152215.1152266

[3] Patrick Baudisch and Ruth Rosenholtz. 2003. Halo: a technique for visualizing off-screen objects. In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '03). Association for Computing Machinery, New York, NY, USA, 481–488. https://doi.org/10.1145/642611.642695

[4] Stefano Burigat, Luca Chittaro, Visualizing references to off-screen content on mobile devices: A comparison of Arrows, Wedge, and Overview + Detail, Interacting with Computers, Volume 23, Issue 2, March 2011, Pages 156–166, https://doi.org/10.1016/j.intcom.2011.02.005

[5] Fitts, P. M. (1954). The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology, 47*(6), 381–391. https://doi.org/10.1037/h0055392