# COMPARING SELECTION TYPE: CLASSIC POINTING VERSUS NEAREST NEIGHBOR

Erika Garces 118047
erika.patricia.garces.fernandez@uni-weimar.de

Anton Kluev 118026
anton.kliuyeu@uni-weimar.de

Aziz Niyazov 119029
aziz.niyazov@uni-weimar.de

***Abstract*—Technology is changing every day; new devices and ways of interacting with them are being implemented. Unfortunately, there is very few research on whether classical pointer and other selecting methods like nearest neighbor could perform good on a Desktop/Laptop environment and not only for amusement. We conducted an experiment on the performance Nearest Neighbor selecting technique on a Desktop/Laptop environment. We found that Nearest Neighbor selecting technique does not perform better than classical pointer. This is due to the lack of visible pointer, and it requires improvements on the logic behind Nearest Neighbor to be used in Desktop/Laptop environment.***

## INTRODUCTION

Technology today is growing exponentially, nearly everybody has a computer; desktop, laptop, and everybody has gotten used to Classical Pointer manipulated by a mouse or trackpad. However, technology like AppleTV by Apple, and FireTV by Amazon are becoming more popular, and these technologies use another way to manipulate objects on the screen; where there is no graphic pointer shown on the display, and the user is aware of its position by the highlighted object.

This new selection method is becoming more common and popular. Therefore, we want to evaluate whether this selecting technique can be replicated on desktop environment with higher efficiency than the current classical pointer.

## RELATED WORK

Computers were designed to reduce the time humans take doing a task, and by changing or tweaking some setting users can save seconds on those tasks. Over a long period of time those second become hours, here is where the higher performance and increase of efficiency can be seen.

Navigating and selecting a target needs to be easy and comfortable to the user but at the same time it needs to have few actions and gestures [1]. There are some companies researching on that field and have already implemented this adjustments to their system.

In Windows by Microsoft is implemented a function called "snap to default button". This function allows users to access quickly the buttons when navigating programs. By enabling this function, the mouse pointer will automatically snap to default button on a dialogue box that pops up. This approach is quite similar to the one in our research, since the cursor is already at needed position, there is no waste time locating a cursor. We are referring to Fitts Law [2].

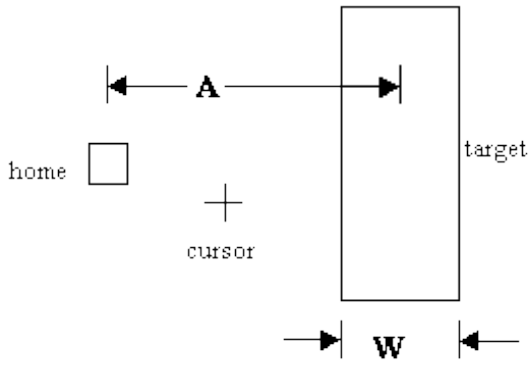


Figure 1. The basic pointing task variables A and W [3]

Fitts Law measures the average time to complete a movement, where:

$$MT = a + b\log_2(A/W + 1)$$

- a and b are constants that depending on input device, and determined empirically by regression analysis.
- A is the distance from the starting point to the target.
- W is the width of the target.
- b is a constant which is either 0, 0.5, or 1, depending on the specific environment.
- Index of difficult (ID) is task difficulty, defined by $\log_2(A/W + 1)$

The "snap to button" tool will take almost no time, making the productivity of a task very high. Distance A is 0 and we took b = 1 to avoid negative index of difficulty, the $\log_2(1)$ is equals 0, which means time required to perform a task is only equals to a, which depends on the input device itself.

Similar approaches had been adopted by other companies; for example, Sony PSP Joystick has a slow

interaction with the screen content. The ability to use such elastic controllers allow us to use only orthogonal layouts.


Figure 2. Sony PSP [4]

Apple, introduced such approach in remote controls of AppleTVs, but also allowing diverse types of layout. Since, AppleTV remote has a trackpad, diagonal movement can be implemented. This interaction is the same that Sony uses in PSP Joysticks, but with a more complex logic behind it. To reduce time to navigate through content, Apple uses the approach on a pointing device that looks similar to the Nearest Neighbor method.


Figure 3. AppleTV [5]

The remote control on Apple TV is a touchpad with buttons. The user can swipe the trackpad in difference position providing him with different actions [3]. Besides, the force or momentum of the swipe can jump over items easier.

Trackpads are a combination of elastic controllers and classical pointing devices, they are more comfortable to use as TV remotes. Elastic controllers such as joysticks are limited in performance, and they are not convenient to be used with classical pointing device, like a regular mouse with a cursor.

Our implementation for Nearest Neighbor has a less complex navigation logic enabling diagonal movements. By using bigger touchpad, momentum is not needed anymore since there is enough area to navigate. In our tested scenario, momentum can be counterproductive and distracting to the user because users are used to an one to one interaction on larger trackpads.

## RESEARCH QUESTION AND HYPOTHESES

The aim of our empirical project is to evaluate the performance of nearest neighbor selecting method in desktop environments. The main focus on of our study was on selecting objects on a touchpad mouse using the nearest neighbor technique. We wanted to focus on a young group of participants; between the age of 18 and 35, since these people are less reluctant to adopt new technologies. The main research question of our project was: Does the usage of nearest neighbor selecting method improve selection performance on desktop environment?

After outlining the research question, we determined our hypotheses. Due to the lack of adoption of new technology, we were expecting the results to move towards one direction; therefore, we decide on one-tailed t-test for evaluation. Thus, we set the following hypotheses:

*H0* : Nearest neighbor method on a desktop environment performs equivalent or worse than Classic pointer.

*H1* : Nearest neighbor method is more efficient in desktop environment.

## DATA COLLECTION

Having in my mind our research question and using our hypotheses, we outlined the variables and testing procedures. Since our emphasis was on measuring performance on selecting techniques; performance is a latent variable, we needed to understand what was performance to determine the correct variable to measure. Performance in our study is referred to shorter time and higher accuracy while selecting objects. The dependent and independent variables are listed in Table 1. We wanted to vary the selecting methods; Nearest Neighbor and Classical Pointer, on a laptop environment, count the number of errors and measure the selection speed. The speed was calculated by measuring the time the participants needed to select all the numbered tiles on the application for each selecting technique. The errors were count once the user clicked an incorrect number or skipped a number on the sequence.

| Independent | Dependent |
|---|---|
| Selection technique | Task competition time in seconds |
| | Number of errors |

Table 1. Independent and Dependent Variables

The data was collected by our Electron Desktop Application for MacOS, and we made use of SurveyMonkey to conduct or questionnaire at the end of the study. The idea behind it was to let the user get to know and be comfortable with both selecting technique, and then choose his preference.

Due to the nature of our experiment, there are confounding variables. The most noticeable confounding variable in our experiment are:

- User skill set regarding trackpads
- Learning curve
- External or environmental effects

We tried minimizing and reduce negative effects of these confounding variables as much as possible by creating a learning session; where the user was able to experience the two techniques and get used to them. Also, the experiment was done on the same laptop, using the same trackpad for all participants. This might had benefit the spent time per task for some users; however, no user complained about the physical device.

Within subject testing was the best approach for our study because we could compare the same group of people using these two techniques. Another perk of within subject testing is that allowed us to gather more data in a shorter period of time; i.e. instead of gathering 60 users, we only needed 30 participants.

Our interval validity was ensured by controlling and managing the testing process; e.g. desktop environment and mouse, and providing the same information to all the participants. External validity was largely guaranteed by the sample size used, there was a cultural diversity among participants.

## EXPERIMENT PREPARATION

A custom Desktop Application based on Electron framework with JavaScript programming language was implemented to test our hypotheses and record the results This application allowed us to track the exact time from the first click to the last, sequential error rate of each individual participant. Additionally, we made a survey to gather general information about the participants, as well as some subjective feedback about the conducted tests.

In order to keep this experiment simple we decided to use only one independent variable that compares two approaches on the randomized layout. In some cases, where the targets are placed around the cursor, the classical pointer might perform worse in comparison with the nearest neighbor. This is because the proposed selection technique acts like a radial menu (selection by the angle) where the angular speed beats the distance. This statement needs a deeper research, and it is not covered on this paper.

### *DESKTOP APPLICATION*

We developed an Electron Desktop Application for MacOS, that consisted of two components - a server and renderer process. In our case the renderer process was an user interface that the participant were interacting with. The UI consists of two approaches (screens or modes) - classical mouse navigation and nearest neighbor with a hidden cursor navigation.

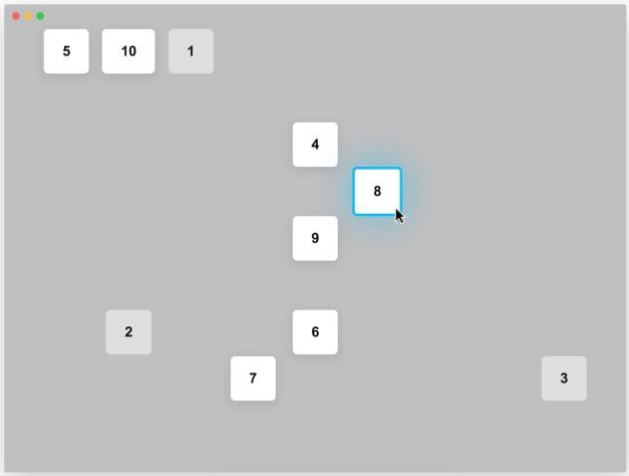


Figure 4. Classical Pointer Layout

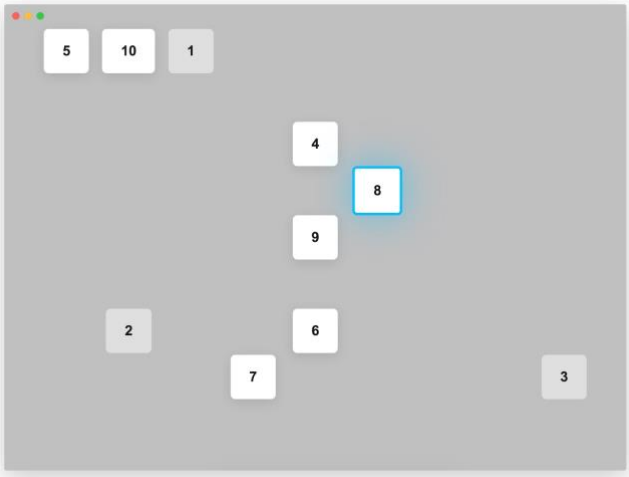


Figure 5. Nearest Neighbor Layout

We used a MacBook multi-touch trackpad in order to mimic the AppleTV remote that uses the one finger trackpad too.

Each screen is filled with randomly distributed, numbered; from 1-10, targets (rectangles) of the same size. For the distribution of a target we have used a 10 x 10 grid and a build-in JavaScript's Math.random() function to define its row and column. In the classic approach the selection process is straightforward. The nearest neighbor required calculation of the nearest target relative to the cursor. On the touchpad-click the target is marked as selected. The cursor in the nearest neighbor technique is hidden to mimic the navigation like in AppleTV.

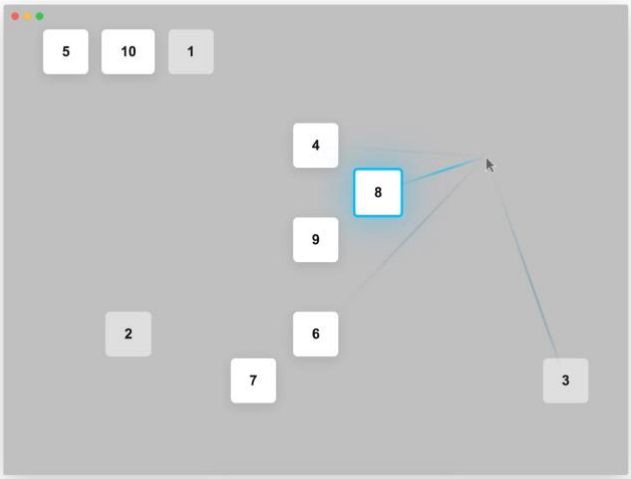


Figure 6. Nearest Neighbor hypothetical cursor position

At the beginning of every test cycle, the application has a list with target id's. On every user selection we check its position in the list. If it is not on the first position, then the user changed the order of selection, and we counted it as an error or a slip. Afterwards we delete the id from the list for the next user selection. This approach allowed to track the correct ascending order.

The Application has also hotkeys to switch approaches manually, reset the tries, and toggle between trial and data gathering. While the introduction into the experiment the application is in the "safe" mode and don't gather the data while the participant gets familiar with the navigation. After the user is ready, the supervisor switches the apps state to aim and the user has one try in each navigation approach. At the end of the first approach the application goes automatically to the next one.

After both approaches are done the renderer process combines data into a JSON struct:

```
10192049583245: {
        pointer: {rr: 0, time: 10.435},
        nearest: {err: 2, time: 11.310}
        }
```

The root key of the struct is the random user id, and its children are the two different approaches, that contains errors and total completion time measured from the first to the last click. At the end of the experiment the JSON struct is sent to the server which gathers the data in a form of a JSON string to the file on HDD.

***QUESTIONNAIRE***

In order to understand who is our test person we created a questionnaire. The last stage of our experiment. It contained four basic questions: gender (male, female), age (18-25, 26-35, 36-45, 46 and more), background (IT & Science, Art & Design and Management) and preferred selection type (Classical Pointer vs. Nearest Neighbor).

## CONDUCTING EXPERIMENT

The study was carried out over a span of a week. Random participants were selected by approaching them in the Bauhaus University Weimar. We informed them about the main goal of the study, and after they had provided their consent the experiment was started. The study was done using the same laptop for all the participants, to decrease confounding variable due to familiarity of their own mice.

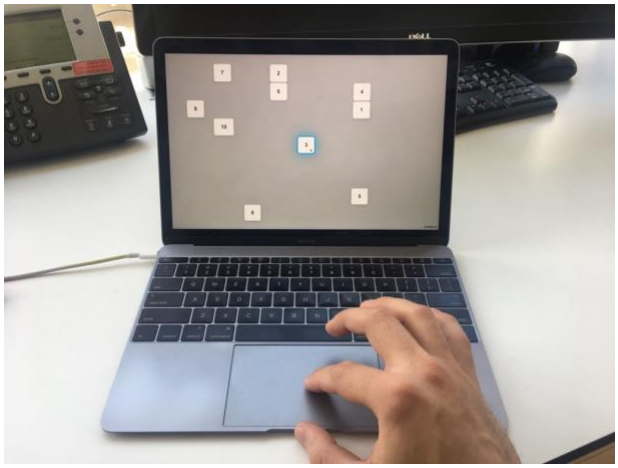

Figure 7. User participating on experiment

First, it was set the Desktop Application; afterwards, it was explained to the tester the aim of the research was about performance of selecting techniques. It was also mentioned that the participant could stop at any given time the experiment. Before starting the test, it was given the instruction to the users and provided the learning setup, so he could get familiarized with the device and the

environment. During the learning phase questions were answered. After this phase was finalized, the first of the two tests was started. For each the position of the tile was randomly distributed in the screen space. Once the participant was done with the first part, the settings changed to the second test. At the end, the user was asked to fill out the Survey Monkey questionnaire.

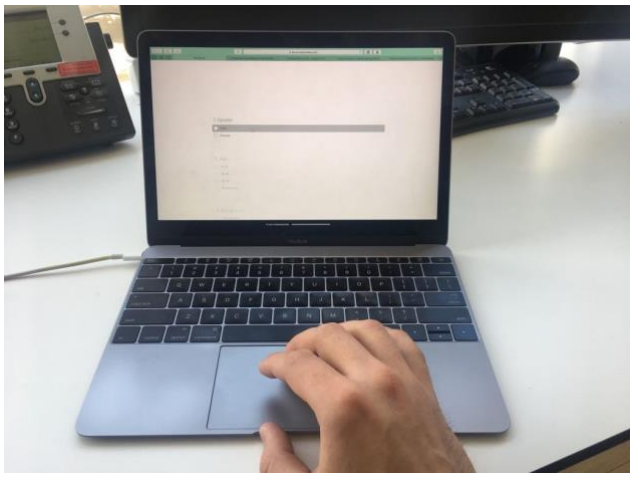

Figure 8. User answering survey

Subsequently, a short discussion was held with the participants to hear their feedback about the study selecting technique. In case that any participant would had had any severe problem, the study would have been stopped and the data discarded.

## DATA EVALUATION

The test was conducted by 31 participants, in an span of a week. The results from the developed application were saved on a JSON files, and it was not required to change it to any particular format because the statically analysis was done using Python3, and modules of Stats and NumPy.

### *PARTICIPANTS*

In our experiment we had a total of 31 participants; 22 male and 9 female, with an age range between 18 and 35 years old. The age was divided in ranges, from 18 to 25 and 26 to 35, 42% of our participants were in the first category while 58% were in the latter. All of the participants work or study at Bauhaus University Weimar. Most of our tested subjected had IT or science background, only a 30% came from arts and design faculties. The same ratio was shown when we asked the participant about their preference on selection techniques. 70% chose Classical Pointer while 30% selected Nearest Neighbor.

### *APPLICATION*

There was a very drastic difference between the two selecting methods. The boxplots on Figure 9 show the clear difference between these methods. Nearest Neighbor had $\mu$ = 15.6 seconds and a $\sigma$ = 5.95, while Classical Pointer had $\mu$ = 11.45 seconds and a $\sigma$ = 2.87.

Additionally, the number of committed errors were less in Classical Pointer than in Nearest Neighbor. Even though, the mean for both technique is the same the standard deviation is slightly larger in Classical Pointer by 1.018, while for Nearest Neighbor is 0.7. The maximum number of committed errors in Classical Pointer was 5 while in Nearest Neighbor was only 3. However, 25 % participants had errors using Nearest Neighbor method, and only a 16% committed errors using Classical Pointer.

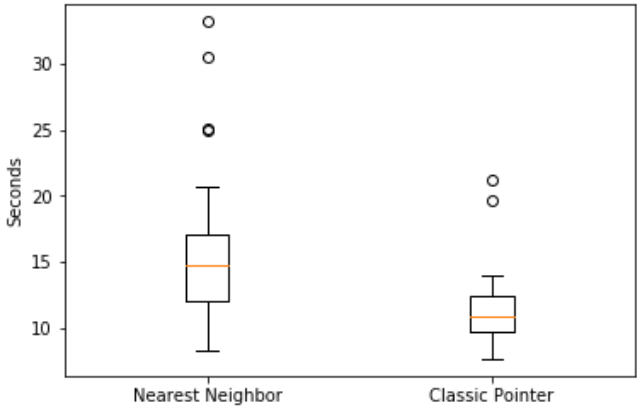


Figure 9. Boxplot for time spent in each method.

We wanted to check whether the data was normally distributed, we conducted the Shapiro Wilk Test. On table 2, the results of the test are stated. Our data was not normally distributed; hence, we reject our null hypothesis that the data is normally distributed . On figure 10 and 11, it is shown the histogram for both methods.

| Technique | Shapiro Wilk Test |
|---|---|
| Nearest Neighbor | p-value = 0.001153 |
| Classical Pointer | p-value = 0.000130 |

Table 2. Shapiro-Wilk Test results

The excess kurtosis on Classical Pointer is 4.1574; therefore, it is a leptokurtic distribution with a Skewness of 1.88, given than the skewness is greater than one we can assumed that the data is highly skewed.

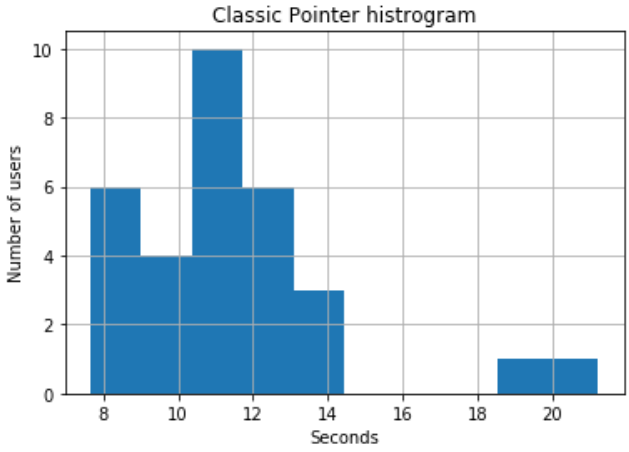


Figure 10. Classical Pointer histogram

The Nearest Neighbor is a dataset with Positive Skewness of 1.384; this typically means that the right-hand tail is longer than the left-hand tail, and it has a kurtosis of 1.643.

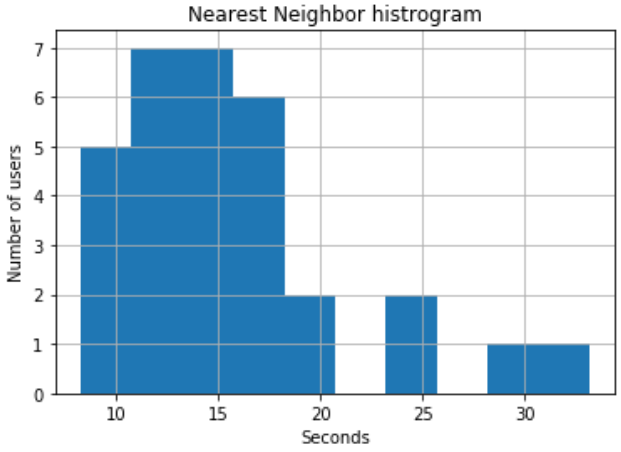


Figure 11. Nearest Neighbor histogram

Given that our data was not normally distributed, we needed to do a Mann-Whitney U test because it is the nonparametric equivalent for t-test. We had an intuition that using Nearest Neighbor would increase the times per task and increase the number of errors. On table 3 is shown our results, and our P value is greater than 5%; therefore; we failed to reject the null hypothesis.

| Technique | Mann-Whitney U Test |
|---|---|
| Nearest Neighbor | statistic=738.0<br>p-value=0.999 |
| Classical Pointer | |

Table 3. Mann-Whitney U Test results

## INTERPRETATION

From our experiment data in the Desktop application is was pretty clear that there was a significant difference between Classical Pointer and Nearest Neighbor method, but our main hypothesis was that to prove that Nearest Neighbor performed better than Classical Pointer. The test data supported that we fail to reject our null hypothesis, therefore Nearest Neighbor performs the equal or worse than Classical Pointer. This can be seen in our Figure 9, where the boxplots for both methods are shown, and it is clearly demonstrated that Nearest neighbor approach takes longer than Classical Pointer.

Analyzing our survey data, we were able to find a patterns between the users background and their preference. Most of the user that had an IT and Science background tend to prefer a classical pointer, while Art and Design student preferred Nearest Neighbor approach. This assumption needs to be study in depth, and it was not covered in our study. We think that it is very interesting to find these types of patterns and taste in people towards an specific type of selecting technique. If future research is done in this direction, and this assumption comes to be true, then specific software and tools for Art and Design student could have Nearest Neighbor interaction.

## CONCLUSION

In this research we failed to reject the null hypothesis. That means, that the nearest neighbor technique performs equal or worse than a classical pointer approach. This study is our first attempt to improve the classical pointer selection technique with usage of the nearest neighbor logic with a hidden cursor. Our technique has a lot of room for improvement in use on desktop computers. In order to simplify the selection logic we made the user harder to orient in the 2D space. Because the nearest neighbor requires the movement of the cursor more than the halfway between A and B targets, it still require some precision and (B-A) * 0.5 distance to drag the finger. It might be the reason, why AppleTV has a complex logic implemented in the selection engine. The next step might be to improve our selection engine and taking the AppleTV selection logic into consideration to minimize user precision effort and finger drag distance.

## REFERENCE

[1] *Human Interface Guidelines* . (2018, March). Retrieved from Apple Developer: https://developer.apple.com/tvos/human-interface-guidelines/app-architecture/navigation/

[2] *Fitts's law*. (2018, March). Retrieved from Wikipedia : https://en.wikipedia.org/wiki/Fitts%27s_law

[3] Goktürk, M. (2018, March). *Fitts's Law*. Retrieved from The Glossary of Human Computer Interaction : https://www.interaction-design.org/literature/book/the-glossary-of-human-computer-interaction/fitts-s-law

[4] *Консоль портативная SONY PSP-E1008/CB Tekken: Dark Resurrection*. (March, 2018). Retrieved from ZAO Patio : https://5element.by/products/499191-konsol-portativnaya-sony-psp-e1008-cb-tekken-dark-resurrection

[5] *How To Update to the Latest the Apple TV Operating System*. (2018, March). Retrieved from Lifewire: https://www.lifewire.com/update-latest-apple-tv-os-1999697