Comparative Study of Clustering Techniques on Eye-Tracking in Dynamic 3D Virtual Environments

Scott Johnson
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/etd

Part of the Computer Sciences Commons

Recommended Citation
Johnson, Scott, "Comparative Study of Clustering Techniques on Eye-Tracking in Dynamic 3D Virtual Environments" (2023). All Graduate Theses and Dissertations. 8885.
https://digitalcommons.usu.edu/etd/8885
COMPARATIVE STUDY OF CLUSTERING TECHNIQUES ON EYE-TRACKING IN DYNAMIC 3D VIRTUAL ENVIRONMENTS

by

Scott Johnson

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

Computer Science

Approved:

Soukaina Filali Boubrahimi, Ph.D.
Major Professor

Isaac Cho, Ph.D.
Committee Member

Brent Chamberlain, Ph.D.
Committee Member

D. Richard Cutler, Ph.D.
Vice Provost of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2023
ABSTRACT

Comparative Study of Clustering Techniques on Eye-tracking in Dynamic 3D Virtual Environments

by

Scott Johnson, Master of Science
Utah State University, 2023

Major Professor: Soukaina Filali Boubrahimi, Ph.D.
Department: Computer Science

Eye-tracking has been used for decades to understand how and why an individual focuses on particular objects, areas, and elements of space. A vast body of knowledge exists on how eye-tracking is measured. However, historically, eye-tracking has been predominately studied using 2D environments, with limited work in 3D environments. The purpose of this study is to identify which methods most accurately represent the areas that have captured the participant’s visual attention within a 3D dynamic environment. This will be completed by evaluating different clustering methods of fixations using a customized virtual reality tool that collects eye-tracking data. There exist several different clustering techniques that could result in varying representations of fixation phenomenon. Thus, selecting the most appropriate clustering algorithm for different eye-tracking datasets is vital. This leads us to the problem of interest. We expect that traditional methods of clustering may fall short in this new scenario.

This thesis will conduct a comparative analysis of several clustering methods. These include DBSCAN, OPTICS, BIRCH, Affinity Propagation, and Mean Shift. These methods range greatly in complexity and provide emphasis on different aspects of the data to ensure we find the most appropriate method(s). To test the appropriateness of each method, we
will create four scenarios, each with a known number of distinct targets. Each scenario will be increasingly complex, ranging from static to dynamic. We task participants to look at the targets when they appear and for as long as they appear. Eye-tracking data on those targets will then be used as input for the clustering methods. The accuracy of each method will be measured by how well it represents the correct number and location of the known targets. An adequate number of participants will be used to produce the necessary data.
Comparative Study of Clustering Techniques on Eye-tracking in Dynamic 3D Virtual Environments
Scott Johnson

Eye-tracking has been used for decades to understand how and why an individual focuses on particular objects, areas, and elements of space. A vast body of knowledge exists on how eye-tracking is measured. However, historically, eye-tracking has been predominately studied using 2D environments, with limited work in 3D environments. The purpose of this study is to identify which methods most accurately represent the areas that have captured the participant's visual attention within a 3D dynamic environment. This will be completed by evaluating different clustering methods of fixations using a customized virtual reality tool that collects eye-tracking data. There exist several different clustering techniques that could result in varying representations of fixation phenomenon. Thus, selecting the most appropriate clustering algorithm for different eye-tracking datasets is vital. This leads us to the problem of interest. We expect that traditional methods of clustering may fall short in this new scenario.

This thesis will conduct a comparative analysis of several clustering methods. These include DBSCAN, OPTICS, BIRCH, Affinity Propagation, and Mean Shift. These methods range greatly in complexity and provide emphasis on different aspects of the data to ensure we find the most appropriate method(s). To test the appropriateness of each method, we will create four scenarios, each with a known number of distinct targets. Each scenario will be increasingly complex, ranging from static to dynamic. We task participants to look at the targets when they appear and for as long as they appear. Eye-tracking data on those targets will then be used as input for the clustering methods. The accuracy of each method
will be measured by how well it represents the correct number and location of the known targets. An adequate number of participants will be used to produce the necessary data.
To my cute wife and adorable kiddo.
PREFACE

This project is supported by the US Army Research Institute of Behavioral and Social Sciences (award: W911NF2010291). The views, opinions, and/or findings contained in this report are those of the authors and shall not be construed as an official Department of the Army position, policy, or decision unless so designated by other documents.
ACKNOWLEDGMENTS

Here I would like to acknowledge the team who has contributed to this work. As the lead author of this thesis, I have conducted the vast majority of coding, created the entire virtual environment, and produced the overwhelming amount of text. I will acknowledge other team members on this project using Elsevier’s CRediT statement (https://www.elsevier.com/authors/policies-and-guidelines/credit-author-statement).

Scott Johnson: conceptualization, methodology, software, validation, formal analysis, investigation, data curation, writing (all activities), visualization.

Dr. Brent Chamberlain: Supervision, Funding acquisition, Writing - Review & Editing.

Charisse: Software.

Sarah: Conceptualization, Methodology.

Jeanine: Conceptualization, Methodology.

I would also like to specifically thank my supervisor Professor Brent Chamberlain for the exceptional support and guidance throughout this research. I would also like to thank all of those working in the VIVID lab and those at the University of Utah. Thank you for your help, support, and great times!

Scott Johnson
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT</td>
<td>iii</td>
</tr>
<tr>
<td>PUBLIC ABSTRACT</td>
<td>v</td>
</tr>
<tr>
<td>PREFACE</td>
<td>viii</td>
</tr>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>ix</td>
</tr>
<tr>
<td>LIST OF TABLES</td>
<td>xii</td>
</tr>
<tr>
<td>LIST OF FIGURES</td>
<td>xiii</td>
</tr>
<tr>
<td>ACRONYMS</td>
<td>xiv</td>
</tr>
<tr>
<td>1 INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1 What is Eye-Tracking and How is it Used?</td>
<td>2</td>
</tr>
<tr>
<td>1.2 Important Eye Movement Definitions</td>
<td>2</td>
</tr>
<tr>
<td>1.3 The Shift from 2D to 3D Eye-tracking</td>
<td>3</td>
</tr>
<tr>
<td>1.4 What is the Existing Knowledge?</td>
<td>4</td>
</tr>
<tr>
<td>1.5 Research Questions:</td>
<td>5</td>
</tr>
<tr>
<td>2 DATA COLLECTION, DATA ANALYSIS, AND FEATURE SELECTION</td>
<td>7</td>
</tr>
<tr>
<td>2.1 Introduction to the Dataset</td>
<td>7</td>
</tr>
<tr>
<td>2.2 Data Acquisition</td>
<td>8</td>
</tr>
<tr>
<td>2.3 Scenarios for Data Collection</td>
<td>9</td>
</tr>
<tr>
<td>2.3.1 Scenario 1: Static Participant, Static Targets</td>
<td>10</td>
</tr>
<tr>
<td>2.3.2 Scenario 2: Static Participant, Moving Targets</td>
<td>10</td>
</tr>
<tr>
<td>2.3.3 Scenario 3: Moving Participant, Static Targets</td>
<td>11</td>
</tr>
<tr>
<td>2.3.4 Scenario 4: Moving Participant, Moving Targets</td>
<td>11</td>
</tr>
<tr>
<td>2.4 Data Preprocessing and Feature Selection</td>
<td>12</td>
</tr>
<tr>
<td>3 METHODS AND EXPERIMENTS</td>
<td>12</td>
</tr>
<tr>
<td>3.1 Performance Metrics</td>
<td>12</td>
</tr>
<tr>
<td>3.2 Algorithms</td>
<td>13</td>
</tr>
<tr>
<td>3.2.1 DBSCAN</td>
<td>13</td>
</tr>
<tr>
<td>3.2.2 OPTICS</td>
<td>13</td>
</tr>
<tr>
<td>3.2.3 BIRCH</td>
<td>14</td>
</tr>
<tr>
<td>3.2.4 Affinity propagation</td>
<td>15</td>
</tr>
<tr>
<td>3.2.5 Mean Shift</td>
<td>15</td>
</tr>
</tbody>
</table>
4 RESULTS AND DISCUSSION

4.1 Performance Metric Results: Number of Clusters 17
4.2 Performance Metric Results: Cluster Distance Accuracy 18
4.3 Performance Metric Results: Algorithm Runtime 19
4.4 Summary 21
4.5 Discussion 21
4.6 Considerations and Clustering Errors 23
4.7 Limitations 24
4.8 Future Scope 25

5 CONCLUSION AND FUTURE SCOPE 27

5.1 Conclusion 27
# LIST OF TABLES

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>A list of the names and functions of a single data point</td>
<td>7</td>
</tr>
<tr>
<td>4.1</td>
<td>The average error for the number of clusters per algorithm</td>
<td>18</td>
</tr>
</tbody>
</table>
**LIST OF FIGURES**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>A picture of the environment used in the experiments.</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Matrix of Experimental Scenarios</td>
<td>10</td>
</tr>
<tr>
<td>4.1</td>
<td>Static Person Static Targets Distance Errors</td>
<td>19</td>
</tr>
<tr>
<td>4.2</td>
<td>Static Person Moving Targets Distance Errors</td>
<td>20</td>
</tr>
<tr>
<td>4.3</td>
<td>Moving person Static Targets Distance Errors</td>
<td>20</td>
</tr>
<tr>
<td>4.4</td>
<td>Moving Person Moving Targets Distance Errors</td>
<td>21</td>
</tr>
<tr>
<td>4.5</td>
<td>Summary of results</td>
<td>22</td>
</tr>
<tr>
<td>4.6</td>
<td>An additional fixation made by the user not on a target</td>
<td>24</td>
</tr>
<tr>
<td>4.7</td>
<td>Miscalibration causes rays to spill over to the building to the right of the target</td>
<td>25</td>
</tr>
</tbody>
</table>
## ACRONYMS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>VOR</td>
<td>Vestibular Ocular Reflex</td>
</tr>
<tr>
<td>AOI</td>
<td>Areas of Interest Network</td>
</tr>
<tr>
<td>DBSCAN</td>
<td>Density-Based Spatial Clustering of Applications with Noise</td>
</tr>
<tr>
<td>LANDSAT</td>
<td>Land Remote-Sensing Satellite</td>
</tr>
<tr>
<td>OPTICS</td>
<td>Ordering Points To Identify the Clustering Structure</td>
</tr>
<tr>
<td>BIRCH</td>
<td>Balanced Iterative Reducing and Clustering using Hierarchies</td>
</tr>
<tr>
<td>CFT</td>
<td>Clustering Feature Tree</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION

With the rise of VR and AR technologies, we need accurate ways to assess human-computer interactions. An excellent way to assess these interactions is through eye-tracking. Eye-tracking has been used for decades in a variety of research\[1\]. With effective tracking tools and analytics to help organize data, researchers can gain insights into human visual perception and cognition. This particular work is part of a broader project (identified in the Acknowledgements), wherein we want to assess an individual’s perception of gist between different environmental contexts. To accomplish this, we needed a way to observe what kinds of observations individuals make within the environment. Unfortunately, eye-tracking data can be immense and needs to be refined. This thesis aims to study how eye-tracking data can best be gathered and evaluated in a three-dimensional, virtual environment. Further, this thesis attempts to find the best means to cluster that data while also aiding others looking to collect and analyze eye data for their experiments.

Eye-tracking data is immense. Thousands of lines of data are created for mere minutes of tracking. Amongst the multitude of eye-tracking research and classification methods, only a handful assess eye-tracking in 3D virtual environments\[2\], \[3\]. Even fewer of these allow for dynamic situations where objects or people move through the virtual space\[4\]. This seems counter intuitive as the nature of VR is to create environments that parallel real-world simulation. In reality, people are constantly reacting and interacting with the environment. The purpose of this thesis is to assess which classification methods perform best under the conditions presented in different virtual environments; environments where people and/or objects move around similar to real life. We aim to condense this new information into more meaningful and interpretable results. This thesis will serve to help future programmers decide which algorithm to use for their specific type of environment.

In this thesis, we select and evaluate 5 unsupervised clustering methods. Because there
are no datasets available for classified eye-tracking data in virtual 3D environments, we have developed an experiment to collect the necessary data. Participants will don a VR headset where they will be asked to stare directly at virtual targets as they appear in the scene. We will use the location and number of targets as a ground truth, or expected outcome, for the algorithms. The algorithms are evaluated on accuracy and algorithm runtime speed. The accuracy can be broken into two parts: did the algorithm capture the correct number of clusters/fixations? What was the distance between the center of the cluster and the target?

1.1 What is Eye-Tracking and How is it Used?

Eye-tracking is a method in which eye movements are recorded and studied[5]. Most modern eye-tracking devices are video trackers[6]. Video trackers take an image of the eye and compute its angle with regard to the visual stimulus. Eye movements are tracked to understand what, when, and how long the eye is focused. Eye-tracking has been used in a variety of research including medicine, sports, marketing, and more[7], [8].

Several eye phenomena have been studied and documented. Fixations and Saccades are most associated with static stimuli and situations. Other phenomena occur in dynamic stimuli such as vergence eye movements, smooth pursuit, and Vestibular Ocular Reflex (VOR). Further explanation is provided in the next section courtesy of Tobii, a leader in the eye-tracking industry[9].

1.2 Important Eye Movement Definitions

**Fixation**: fixations occur when the eye stops moving to focus on a specific object or scene. This keeps the foveal area of our eye in place to take in detailed visual information. Several studies have been performed to determine fixations frequency. One study by Negi and Mitra[10] involved participants looking at video data. This data showed that most fixations were less than 500ms. However, some fixations lasted several seconds. It is expected that the longer something is fixated on, the more effort is required to visually process the visual information. This information can be pertinent for specific algorithms, possibly putting a soft cap on how big a cluster can be based on how much time it takes
Saccade: Saccades are the eye movements that occur between fixations. Saccades can occur voluntarily or involuntarily and last anywhere from 20ms to 240ms[11]. These rapid eye movements produce images of poor quality, therefore adding little to no visual information. Because of this, we want to discard saccadic gaze points during eye-tracking analysis.

Vergence: This happens when both eyes move in opposite directions. If both eyes are moving towards the nose that is called a far-to-near focus, and the opposite is called near-to-far focus. This helps prevent double vision and is slower than a saccade. For this paper, we are not interested in vergence because it is too difficult to detect past a convergence of three meters.

Smooth Pursuit: Smooth pursuit is when your eyes rotate together to maintain focus on a moving object. It cannot be triggered voluntarily, there must be a moving object to follow for it to be considered “smooth.” For the sake of our experiment, this could pose a problem to some algorithms as the ‘clusters’ of data may be spread in such a way that a cluster would not be formed when it should.

Vestibular Ocular Reflex (VOR): VOR is when the eyes are focused on a specific point, but your head moves. This reflex keeps your eye focused on what you want by moving the eyes at the same rate we move the head. Because our algorithms will not have the head rotation as an input, this phenomenon should not affect our testing and will not be considered.

1.3 The Shift from 2D to 3D Eye-tracking

Most of the research conducted on eye-tracking has been based on 2D static scenes, wherein algorithms do not need to take into account the impact of user or object movements on eye movements. The absence of changes in distance in such scenarios negates the occurrence of vergence, as there is only a single point of focus for the eyes. Smooth pursuit, which occurs when tracking moving objects, is another phenomenon that cannot be voluntarily initiated and is thus absent in static scenes. Therefore, such phenomena
cannot occur or exist in 2D environments. For this reason, we reiterate there is a need to find which algorithms can handle these new events.

Another issue that comes with the transition from 2D to 3D is it allows for even more dimensions to be explored. This can lead to an increase in the curse of dimensionality. The curse of dimensionality refers to the fact that as the number of dimensions or features in a dataset increases, the data becomes increasingly sparse, which makes it difficult to analyze and interpret. This sparsity also leads to increased computational complexity and can cause certain algorithms to break down or become ineffective. Deciding which variable to use as input and how many dimensions are necessary becomes even more imperative.

There have been studies that use dynamic environments such as the work by Ugwitz et al.[4] and Pfeiffer et al.[12]. However, these do not perform any analysis on fixations and saccades and rely mostly on theoretical workflows. Our study, however, will utilize the ground truth inherent in the experiment. The duration and location of intended fixations will be known and can be used to judge the effectiveness of our algorithm.

1.4 What is the Existing Knowledge?

There exists a multitude of prior knowledge on eye-tracking in 2D space. But there are very few papers that explore fixation detection in dynamic 3D virtual environments. As mentioned in the previous section, a notable paper written by Ugwitz et al.[4] discusses several points tangential to this work. This paper evaluates different solutions to eye-tracking in interactive virtual environments. However, these were done on a theoretical basis and not through practical application. The paper focuses on a workflow and software architecture for an entire experimental scenario. This includes virtual scene preparation, experimental data collection, considerations for ambiguous visual stimuli, post hoc data correction, data aggregation, and visualization.

The paper titled Bayesian Online Clustering of Eye Movement Data[13] proposes a learning algorithm that can detect fixations in real-time. As the visual scene changes and new objects appear, based on a mixture model, the algorithm can identify and tell visual saccades from visual fixation clusters. The approach is evaluated on real-world data, col-
lected from eye-tracking experiments in driving sessions. What is especially interesting about this paper is that it has somewhat of a modified 3D experience. Though participants were looking at a 2D screen, the video shows a car moving through 3D space with fixations correlating to objects visible at the time in the video.

Salvucci et al. propose a taxonomy of fixation identification algorithms in their paper titled Identifying Fixations and Saccades in Eye-Tracking Protocols[14]. The authors classify algorithms based on how they use spatial and temporal information in eye-tracking protocols. The authors describe five representative algorithms from different classes in the taxonomy that are based on commonly used techniques. The algorithms are then evaluated and compared against several qualitative characteristics. The authors found that velocity-based and dispersion-based algorithms perform equally well. However, area-based algorithms are too restrictive and can generate biased results. Building on this work, this thesis focuses the input of our data to provide both velocity and dispersion-based information.

Lastly, the paper titled Space-Time Visual Analytics of Eye-Tracking Data for Dynamic Stimuli[15] introduces a visual analytics method for analyzing eye movement data that is recorded for dynamic stimuli, such as video or animated graphics. The method focuses on analyzing data from multiple viewers to identify trends in general viewing behavior, including time sequences of attentional synchrony and objects with strong attentional focus. The authors use a space-time cube visualization in combination with clustering to analyze the dynamic stimuli and associated eye gazes in a static 3D representation. Shot-based, spatiotemporal clustering of the data generates potential areas of interest that can be interactively filtered. They used these data to find Areas of Interest (AOIs) in the videos and scenes and found that moving objects almost always garnered the most attention.

1.5 Research Questions:

Question 1. : Given the data we collected, which algorithm performs the best given our criteria (Accuracy and algorithm runtime)?

Question 2. : Do different algorithms perform better in different virtual scenarios
than others? Specifically, will some algorithms be able to adapt to changes in movement throughout the scene better than others?
CHAPTER 2
DATA COLLECTION, DATA ANALYSIS, AND FEATURE SELECTION

This section will introduce the dataset we will be working with. This includes data that will be collected but not necessarily used in the clustering process. Following the dataset, we will explain the environments we will be using to obtain our data along with the procedures of the experiment. Finally, we will discuss which features of the data will be used as input for our clustering algorithms.

2.1 Introduction to the Dataset

The dataset consists of just under 60 sets of eye-tracking data. Each set of data represents the gaze patterns for one participant and one scenario. Each participant goes through 4 different scenarios and produces a set for each. Therefore, we have 4 groups of 20 sets for each of the different scenarios. The features we are collecting are more than we need for our purposes, this is because we are utilizing an eye-tracking system already made in our lab for other research purposes.

Table 2.1: A list of the names and functions of a single data point

<table>
<thead>
<tr>
<th>Time</th>
<th>The time at which the point of data was collected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Name</td>
<td>The name of the object the participant was looking at.</td>
</tr>
<tr>
<td>Collision object position</td>
<td>The X, Y, and Z coordinates of the object being looked at.</td>
</tr>
<tr>
<td>Coordinate of ray collision</td>
<td>The X, Y, and Z coordinates of where the eye was looking.</td>
</tr>
<tr>
<td>HMD position</td>
<td>The X, Y, and Z coordinates of the head in virtual space.</td>
</tr>
<tr>
<td>HMD rotation</td>
<td>The rotation of the head in virtual space.</td>
</tr>
<tr>
<td>Gaze direction</td>
<td>The direction in which the virtual ray was cast from the eye.</td>
</tr>
<tr>
<td>Eye openness</td>
<td>An indication of how open the eye is. Used to assess blinking.</td>
</tr>
</tbody>
</table>
Each set of data consists of eight features: Time, object name, the coordinate of ray collision, collision object position, HMD position, HMD rotation, gaze ray direction, and eye openness. Table 2.1 shows a list of the names and functions of a single data point. The frequency that the eye tracker produces data is equal to the refresh rate of the headset. While running the experiment we saw updates occurring approximately 45 frames per second. In total, we collected over 103 thousand points of data for testing.

2.2 Data Acquisition

To generate the eye-tracking data we will use, a series of simple environments have been created in Unity. To test the validity of the methods proposed, a series of simple environments will be created using the Unity3D engine and Vive Eye Pro. The experiment will be broken up into 4 parts given to the participant in succession (taking roughly 20 minutes in total). No demographic data will be collected during this process. Each of the following scenarios will be presented to the participant in a random order. The scenarios require the user to use their head and eye movements to find and focus on a bullseye target. Each target is displayed for roughly one second.

Figure 2.1 shows the virtual environment users will be placed in when they put on the headset. Note that in this image all targets are visible, but during the experiment, only one will be visible at a time to the user. The four scenarios used in the experiment will have the following in common:

- Each target will be the same: A classic red and white bullseye.
- Half a second before a target will expire, the next target will be displayed. This time is for the participant to switch gaze to the next target and avoid creating a false fixation behind a recently deleted target.
- A target will appear in the field of view of a disappearing one. In other words, participants will not have to search for a target outside of their peripheral vision when the current one disappears.
Fig. 2.1: A picture of the environment used in the experiments.

- Each environment will contain identical buildings and a road. This will help maintain a virtual presence for the participant and help avoid VR sickness.

- Targets will appear for roughly one second with some variation to prevent anticipatory glances.

- There will be adequate time and space between targets so quick reflexes will not be required to follow the targets.

- The order in which the scenarios appear will be random.

2.3 Scenarios for Data Collection

The subsequent sections explain each scenario used in the data collection process. The four scenarios are a systematic way to test the types of scenarios you might encounter in 3D virtual environments. Table 2.2 outlines the different scenarios.
2.3.1 Scenario 1: Static Participant, Static Targets

This scenario is the most basic and straightforward of the scenarios. The participant will remain stationary throughout the experiment. When targets appear, they will be stationary as well. With this experiment, we hope to gather very specific, easy-to-cluster data. We anticipate that most clustering models will have little trouble determining the number, position, and duration of fixations.

2.3.2 Scenario 2: Static Participant, Moving Targets

This scenario steps it up a bit and adds movement to the targets. When the target is created, it will start to move towards another position slowly. The goal is that it is not too fast to follow with your eyes. With this scenario, we hope to gather some smooth pursuit data. This will create a bit of a challenge for some of the algorithms because fixations should be spread out along a line.

2.3.3 Scenario 3: Moving Participant, Static Targets

This scenario adds movement only to the participant, the targets remain stationary. Each participant will be moving through the virtual space at nine meters per second. In this scenario, we expect to see more eye movement, even during fixation, but the gaze points should make nice clusters for evaluation.
2.3.4 Scenario 4: Moving Participant, Moving Targets

This scenario combines the previous two scenarios and has both the targets and the participant moving. This scenario is the most complex and will be the most difficult to cluster. The algorithms will have to juggle a moving gaze ray, a moving target, and a moving head position. All of which could be happening during a fixation.

2.4 Data Preprocessing and Feature Selection

Each set of data collected will be processed to remove abnormalities. Examples of abnormalities are loss of eye-tracking capabilities, moments when eyes are completely closed, or when the gaze ray does not collide with anything. If an abnormality does occur, it will be manually or automatically removed.

A few features of the data will be updated during this process. The ‘time’ feature is composed of the hour, minute, second, and millisecond of the day that the data was collected. To make this easier for algorithms to perform distance measures, the ‘time’ feature will be converted to the total milliseconds since the scenario began. A similar preprocessing technique will be applied to the gaze direction. This will be converted to the difference in angles from the previous frame to the current frame. For this experiment, we will be using the coordinates of ray collision, gaze direction change, and time as inputs for our algorithms.
CHAPTER 3
METHODS AND EXPERIMENTS

As we have now developed a thorough understanding of the data and eye-tracking, we wish to use different algorithms to separate the data into fixations and saccades. This chapter will discuss the different algorithms used in our experiments along with the metrics we will be using to evaluate the performance of those algorithms. Each algorithm used in this thesis is widely used for clustering data. The chapter will be structured as mentioned below. To evaluate the performance of the algorithms under these conditions, we have come up with 3 metrics: The number of clusters, the accuracy of the clusters, and finally, the algorithm runtime.

3.1 Performance Metrics

Number of Clusters: The first aspect we looked at when assessing the algorithms is if they could match the number of fixations, which we know, with the number of clusters the algorithms produced from the data. For example, if we have 25 targets that appear throughout the scene, then we would expect the algorithm to produce 25 fixations.

Accuracy of Clusters: This measure will tell us how precise the algorithms are at centering the fixation about the gaze destination, in our case the targets. To calculate the center of a fixation, we average the position of each point in a cluster produced by the algorithms. We then take each target position and find the minimum distance to any other fixation cluster center. Those values are averaged and compared to the other algorithms to find the best-performing algorithms.

Algorithm Runtime: Finally, we will measure the time it takes for each algorithm to cluster the data. This is highly important because of the multitude of data that can come from eye-tracking. For example, my experiment took all of 2 minutes of data collection time, others may collect data for over half an hour. If this data takes a long time to process, then
the algorithm would be practically useless in longer scenarios. Each algorithm will receive a score of “Fast”, “Average Speed”, and “Slow”. A “Fast” algorithm performs its clustering in under 10 seconds for our specific dataset. An “Average speed” algorithm produces clusters in over 10 seconds and under one minute. Finally, a “Slow” algorithm will produce clusters in over 1 minute for our specific dataset.

3.2 Algorithms

To find an algorithm that most successfully classifies eye-tracking data, we have selected five clustering algorithms for consideration. These algorithms are DBSCAN, OPTICS, Affinity Propagation, and Mean Shift. Each algorithm has something unique to offer to this problem and we hope to find which one performs best under each of our four scenarios. The following sections describe each algorithm in turn and its uses.

3.2.1 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise)\[16\] is an unsupervised machine learning algorithm used for clustering. It locates dense clusters of points in a data set and groups them. It uses a density-based approach, where points that are close together in the data space are considered to be part of the same cluster, while points that are far apart are considered to be noise or outliers. The algorithm requires two parameters, the radius (\(\epsilon\)) and the minimum number of points (\(\text{minPts}\)) required to form a dense cluster. Points that have at least \(\text{minPts}\) number of points within a radius of \(\epsilon\) are considered core points, and all other points within the radius are assigned to the same cluster. DBSCAN is used in various applications such as pattern recognition, anomaly detection, and data mining.

3.2.2 OPTICS

OPTICS (Ordering Points To Identify the Clustering Structure)\[17\] is a density-based clustering algorithm that is used for identifying and extracting the structure of clusters in large and complex datasets. Unlike traditional clustering algorithms, such as K-means,
which only work well with spherical-shaped clusters, OPTICS is capable of identifying clusters with arbitrary shapes and sizes. This may play a large role in identifying smooth pursuit fixations.

The algorithm works by ordering the objects in a dataset based on their reachability distance, which is a measure of the distance between an object and its nearest neighbors. By constructing the reachability plot, which displays the reachability distances of all objects in a dataset, the structure of clusters in the dataset can be visualized. The user can then determine the number and shapes of clusters in the dataset by adjusting a parameter called the "minimum reachability distance".

OPTICS has several advantages over traditional clustering algorithms, including its ability to handle noise and outliers in the data, its ability to identify clusters with arbitrary shapes and sizes, and its ability to work efficiently with large datasets.

3.2.3 BIRCH

The Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH)[18] algorithm is a hierarchical clustering technique that is particularly well-suited for large datasets. BIRCH operates in two phases: the first phase involves building a hierarchical clustering structure in memory, and the second phase involves using this structure to assign data points to clusters. In the first phase of BIRCH, the algorithm builds a tree-like structure called a Clustering Feature Tree (CFT). The CFT is built using a set of user-defined parameters that control the size of the clusters, the number of subclusters within each cluster, and the maximum number of points that can be stored in each node of the tree. The CFT is built by recursively splitting the data into smaller and smaller clusters until the desired level of granularity is reached. Once the CFT is built, it is used in the second phase of the algorithm to assign data points to clusters based on their distance to the cluster centers stored in the CFT. BIRCH is designed to be both memory-efficient and computationally efficient, making it an attractive option for clustering large datasets.
3.2.4 Affinity propagation

At the heart of affinity propagation[19] is the concept of ”message passing” between data points. Each data point sends messages to other data points to indicate how well it can represent the other points as a representative point for a cluster. These messages are used to update the representative points and cluster assignments until convergence. The representatives are then used to assign data points to clusters.

In affinity propagation, the similarity measure between pairs of points is often computed using negative Euclidean distance. This will be how we measure similarity in our experiment. Affinity propagation is known to be computationally expensive, but it is often able to identify meaningful clusters in high-dimensional datasets where other clustering algorithms struggle.

3.2.5 Mean Shift

Mean shift[20] is a non-parametric clustering algorithm that can be used to identify clusters of arbitrary shape in a dataset. The algorithm works by iteratively shifting points in the dataset towards the direction of maximum increase in the local density of points. The algorithm starts by initializing a set of candidate cluster centers, usually by sampling from the data points or by placing them at regular intervals in the data space. For each candidate center, the algorithm calculates the mean shift vector, which is the vector pointing towards the direction of maximum increase in density. The candidate center is then shifted by this vector, and the process is repeated until convergence is reached. The algorithm assigns each point in the dataset to the nearest cluster center, and the resulting clusters are defined by the set of points that converge to the same cluster center.

Mean shift can be sensitive to the choice of bandwidth parameter, which controls the size of the region used to estimate the density of points around each candidate center. A larger bandwidth can result in more points being assigned to a cluster, while a smaller bandwidth can result in more clusters being identified. Additionally, the algorithm can be computationally expensive for large datasets, since the mean shift vector must be calculated for each data point at each iteration. Nevertheless, mean shift clustering has been success-
fully used in a variety of applications, including image segmentation, object tracking, and computer vision.
In our study, we employed a diverse array of unsupervised clustering algorithms to analyze eye-tracking data to identify fixations. Each cluster that was identified by the algorithms was deemed to be indicative of a fixation that was made by the user. To optimize the performance of the clustering algorithms, we applied moderate tuning to their respective parameters. In the subsequent sections, we will present the evaluation metrics that we utilized to assess the efficacy of these algorithms. Given that we examined four distinct scenarios, we will assess the performance of each algorithm with respect to each metric and scenario.

4.1 Performance Metric Results: Number of Clusters

Upon evaluation of the results and data obtained, we have determined that the number of clusters metric is not particularly valid when considering the accuracy of a clustering method. Even though the numbers on algorithms such as DBSCAN and BIRCH achieve an almost perfect score, further review shows the clusters do not line up with the target positions. Furthermore, each algorithm is highly affected by parameter tuning. It is our educated assumption that each of these algorithms could be tuned per individual to achieve the required number of fixations. For this reason, we have to consider this paired alongside the accuracy of the clusters. However, the results for the average error for the number of clusters are shown in 4.1.

Based solely on the results, DBSCAN is the clear winner in almost every category. Though Mean Shift performs well in the Standing Person Moving Target category, Mean Shift performs quite poorly in the other scenarios.
Table 4.1: The average error for the number of clusters per algorithm

<table>
<thead>
<tr>
<th></th>
<th>DBSCAN</th>
<th>OPTICS</th>
<th>BIRCH</th>
<th>Affinity Propagation</th>
<th>Mean Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>SS</td>
<td>1.285714</td>
<td>4.928571</td>
<td>-7</td>
<td>-2.71429</td>
<td>-5</td>
</tr>
<tr>
<td>SM</td>
<td>0.1</td>
<td>9</td>
<td>-7</td>
<td>2.2</td>
<td>-0.5</td>
</tr>
<tr>
<td>MS</td>
<td>0.75</td>
<td>1.625</td>
<td>-3</td>
<td>-3.5625</td>
<td>8.8125</td>
</tr>
<tr>
<td>MM</td>
<td>2.411765</td>
<td>6.470588</td>
<td>-3</td>
<td>-1.0625</td>
<td>9.705882</td>
</tr>
</tbody>
</table>

4.2 Performance Metric Results: Cluster Distance Accuracy

This measure will tell us how precise the algorithms are at centering the fixation about the gaze destination, in our case the targets. A good algorithm would have a lower error value, indicating clusters closer to the target positions. This is computed by averaging the distances from the targets to the closest fixation. The following four graphs are box plots showing the five-point summary for the data, this includes the mean, median, and interquartile distance.

Looking at the results, we see that DBSCAN and OPTICS consistently score the lowest values for each scenario, while BIRCH, Affinity Propagation, and Mean Shift are generally higher and less accurate. It should be noted that Affinity Propagation performed better in static person environments. This could be because Affinity Propagation was more sensitive to the gaze angle difference.

Further analysis of the data shows that the spread of the data was quite large for DBSCAN and OPTICS. This shows that even though the cluster positions are quite close on average, we have a lot of variability in the results. However, when we look at Affinity Propagation, we see that its variability is very low, except for Moving Person Static Targets. We see consistency that does not appear anywhere else. It should be noted, however, that Affinity Propagation occasionally produced a single cluster for the entire dataset. These data have been removed, but their existence means that some extra parameter tuning may be important for this algorithm.
4.3 Performance Metric Results: Algorithm Runtime

The speed of the algorithm was categorized into three speeds: “Fast”, “Average Speed”, and “Slow”. There were a variety of algorithm speeds used in our evaluation. The results show that DBSCAN and BIRCH are both “Fast” algorithms at 5 seconds and 7 seconds respectively. OPTICS was the only “Average Speed” algorithm at a time of 21 seconds. And finally, the Affinity Propagation and Mean Shift algorithms both performed “Slow” at 84 seconds and 75 seconds respectively. While each algorithm performed at a manageable time for our experiment, other experiments that have more participants and much larger datasets should consider DBSCAN, BIRCH, and possibly the OPTICS algorithm for their clustering methods.
Fig. 4.2: Static Person Moving Targets Distance Errors

Fig. 4.3: Moving Person Static Targets Distance Errors
4.4 Summary

Each algorithm has various pros and cons depending on the environment one wishes to run, and the amount of data produced from those experiments. Given our findings, we are going to make a qualitative list summarizing the results. Each algorithm will receive a score of zero to two checkmarks. A double checkmark is highly performant, and no checkmarks indicate poor performance. The overall category is a loose conglomeration of the three parameters into one.

4.5 Discussion

The objective of this thesis was to find the algorithm that performs the best according to the established performance metrics (research question 1). Based on the results and summary table, it is observed that DBSCAN and OPTICS outperformed other algorithms
Fig. 4.5: Summary of results

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Number of Clusters</th>
<th>Accuracy of Clusters</th>
<th>Algorithm runtime</th>
<th>Overall</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBSCAN SS</td>
<td>✔️✔️</td>
<td>✔️</td>
<td>✔️✔️</td>
<td>✔️✔️</td>
</tr>
<tr>
<td>DBSCAN SM</td>
<td>✔️</td>
<td></td>
<td>✔️✔️</td>
<td>✔️✔️</td>
</tr>
<tr>
<td>DBSCAN MS</td>
<td>✔️</td>
<td></td>
<td>✔️✔️</td>
<td>✔️✔️</td>
</tr>
<tr>
<td>DBSCAN MM</td>
<td></td>
<td>✔️</td>
<td>✔️✔️</td>
<td>✔️✔️</td>
</tr>
<tr>
<td>OPTICS SS</td>
<td></td>
<td>✔️</td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>OPTICS SM</td>
<td></td>
<td>✔️</td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>OPTICS MS</td>
<td>✔️</td>
<td></td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>OPTICS MM</td>
<td>✔️</td>
<td></td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>BIRCH SS</td>
<td>✔️✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>BIRCH SM</td>
<td>✔️</td>
<td></td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>BIRCH MS</td>
<td>✔️</td>
<td></td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>BIRCH MM</td>
<td>✔️</td>
<td></td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>Affinity Propagation SS</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>Affinity Propagation SM</td>
<td>✔️</td>
<td>✔️</td>
<td></td>
<td>✔️</td>
</tr>
<tr>
<td>Affinity Propagation MS</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Affinity Propagation MM</td>
<td>✔️</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean Shift SS</td>
<td>✔️✔️</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean Shift SM</td>
<td>✔️✔️</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean Shift MS</td>
<td>✔️✔️</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Mean Shift MM</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*The number of clusters can be selected beforehand, hence the double checkmark. However, this only applies if you know the number of expected fixations beforehand.*

in most aspects. Considering that DBSCAN yielded slightly better outcomes in the number of clusters, we recommend using it almost exclusively. This leads us to research question 2, which investigates if certain algorithms perform better than others in specific scenarios. Despite the all-around superiority of DBSCAN and OPTICS, Affinity Propagation stands out in one particular case, i.e., when the user is static. However, our tests revealed that
Affinity Propagation is a slow algorithm and should only be considered for small datasets.

4.6 Considerations and Clustering Errors

A significant finding from this experiment is the high occurrence of errors in the collected data. Even when using the most performant algorithms, carefully observing the data often revealed errors and inconsistencies. The following section provides an in-depth examination of the three types of errors that were most frequently observed in the experiments: Human, Hardware, and Algorithmic.

**Human Error**: This error found in our experiments occurred from actions performed by the participant during the experiment. There is little that can be done to mitigate this type of error. Due to the sporadic nature of eye movements and human attention, instances of fixations occurring away from the targets exist. These fixations would be correctly identified by the algorithm(s) but provide an unexpected fixation and throw the accuracy of certain metrics.

An example of human error is shown in Figure 4.6. Given the position of the salmon-colored rays in the middle and the duration they are occurring, an expert would likely classify this as a fixation, even though it was against the instructions for the participant to do this. One possible solution to this problem is increasing the incentive to focus on the targets. Another may be allowing practice sessions to hone one’s skills. Though this may increase the accuracy of this experiment, it is likely impractical for real-world application.

**Hardware Error**: This error is caused by the hardware and computations made by the Vive Eye Pro and its accompanying eye-tracking software. The calibration of the eye position and subsequent ray projected in the virtual scene may have been slightly off. Even a small error can cause fixation algorithms to falter on what a human can see as an obvious fixation. For example, say the participant is looking directly at a far-off target. Though they may be staring intently at the center, the ray direction received from the Vive Eye Pro may be calculated in such a way that the ray does not collide with the target or possibly ‘flickers’ on and off of the edge of the target. An example of this is shown in Figure 4.7.

Though this problem might be fixed with more rigorous calibration, movements during
the experiment may cause the headset to move, throwing off the calibration. Another fix for this type of error would be expanding the collision area beyond the area of the object. Though this may catch those unwanted ‘flickers’ and place them on the target, it may not collect the possibility that the user is simply looking beyond, and adjacent to, the target.

**Algorithmic Error**: This type of error corresponds to an error generated by the algorithm. Simple inaccuracies in the fixations or clusters produced. For example, producing more, or less, fixations than occurred. In these cases, the participants were looking exactly where they were supposed to, and the hardware correctly projected their eye movements onto the target, but the algorithm is to blame for these faulty results. The biggest fix to a problem like this is parameter tuning. Parameter tuning is crucial when applying these algorithms to your data.

### 4.7 Limitations

While this study provides valuable insights into eye tracking data collected within a
virtual environment using the Vive Eye Pro, certain limitations should be acknowledged. Firstly, the sample size used in this study was relatively small, consisting of only 60 sets of data divided among the 4 different scenarios. Furthermore, participants were conveniently selected from the university community. Therefore, caution should be exercised when extrapolating the results to larger populations or diverse demographic groups. Additionally, the study duration of 30 seconds may limit the application of these results. Longer study durations may find other algorithms that perform better or worse under such conditions. Lastly, the data collection frequency of 45 points per second is a relatively small frequency as there are eye tracking devices that can collect at a rate of hundreds of points per second. Finer instruments may require other algorithms to more successfully detect fixations and saccades. Despite these limitations, this study serves as a valuable foundation for future research in understanding eye tracking data analysis in virtual environments.

4.8 Future Scope

In our future research, we would like to change how our algorithms’ performance is evaluated. Instead of the three metrics used, a more complete and thorough validation technique would be generated. Ideally, we would find an expert in eye-tracking and eye movements and use their expertise to label each gaze point as a fixation or a saccade.
Afterward, a confusion matrix could be developed on a per-gaze point basis. We believe this will more adequately answer the question of which algorithm performs best. Earlier we mentioned the curse of dimensionality, finding a different set of inputs might also yield us better results than we currently have.

Finally, each of the errors discussed in section 4.5 would be addressed. For hardware, we could consider a more rigorous calibration step, including instructions to not touch the headset during the experiment or rerunning calibration between scenarios. For human error, there is very little that can be done. More thorough instructions on where to look and perhaps a practice run may assist in more accurate data collection. Finally, algorithmic error, using a different set of inputs may lead us to find what variable can detect when fixations and saccades occur.
5.1 Conclusion

Eye-tracking has been utilized for many years and there exists a vast amount of knowledge regarding its measures and applications. However, this knowledge does not thoroughly dive into dynamic 3D environments. The purpose of this thesis was to identify which methods most accurately represent the areas that have captured the participant’s visual attention within a 3D dynamic virtual environment. To do this, we created an experiment to test different clustering algorithms within different scenarios to find which performed the best. These algorithms were judged on the number of fixations produced, the position of those clusters relative to the targets, and how quickly the algorithm performed the task.

A summary of our findings can be found in section 4.4. Briefly, we can see that DBSCAN had the highest overall score. DBSCAN performed the best for the number of clusters in every scenario. It also performed well with regard to cluster accuracy and runtime performance. This makes it the algorithm of choice and can become even better with more accurate parameter tuning.

The main findings of this study were that eye-tracking data received from the Vive Eye Pro can be messy. Whereas the algorithms may have generated results that appear satisfactory, a deeper look at the visualizations in Unity shows that each result had several errors within it. These errors ranged from Human to Hardware to Algorithmic. Through running this experiment and collecting its results, we have obtained a broader understanding of dynamic 3D eye data. Using this knowledge, we can conclude that there must be a better method for classifying points as fixations and saccades.
REFERENCES


