Team Members: Gentry Atkinson
CS7321: Fall 2019
Project 3 Report

Activity	**Time**
Write Code	Atkinson, 6 hours
Test and Debug Code	Atkinson, 6 hours
Prepare Results	Atkinson, 1 hour
Prepare Slides and Report	Atkinson, 3 hours

1. Introduction
The task of dividing recorded eye movements data into Fixations, Pursuits, and Saccades is fundamentally a problem of grouping data points by similarity. The challenge is to find some expression of the data captured by eye tracking software that allows those groups to present quickly and reliably. Traditionally thresholds have been defined which separate data points by velocity. These techniques are fast and effective but rely on pre-existing knowledge of the data. By contrast, machine-learning based clustering techniques can learn groupings of data based solely on an analysis of the data. For this reason K-Means as been chosen as an experimental technique to classify eye movements for this project. This algorithm is used to divide cleaned data into three groups which are then labeled as Fixation, Pursuit, or Saccade. These labels are compared to ground truth to determine the accuracy of the approach.

2. Background
2.1 K-Means is a clustering algorithm which has been in common usage since 1967. It functions by guessing and then iteratively adjusting k cluster centroids within some data[1]. It is worth noting that K-Means does not try to balance the number of points in each cluster but rather finds centroids in the data the the points group around. K must be defined as an input parameter of the algorithm. Although many newer clustering algorithms have been defined, K-Means allows us to force the finding of exactly 2 clusters which is ideal for the separation of fixations and saccades. More complex algorithms might be better suited to more complex analysis, such as the detection of fixations, saccades, and smooth pursuits. K-Means also offers the advantage of running in quasi- O(n) time which is extremely fast by the standards of machine learning. Clustering algorithms are all very sensitive to the analyst's choice of distance measure. This experiment was conducted using a Euclidean distance measure. The K-Means algorithm is broadly described as:

```
Given set of data points s={$s_1$,...,$s_n$}
Randomly initialize set c={$c_1$,...,$c_k$} of k centroids
Initialize a set of results r={$r_1$,...,$r_n$} to zeros

Loop until c does not change:
  for i = 1 to n:
    min_cluster = 0
    min_distance = MAX_FLOAT
    for j = 1 to k:
      calculate distance($s_i$, $c_j$)
      if (distance < min_distance):
        min_cluster = j
        min_distance = distance
    $r_i$ = min_cluster

  for i = 1 to k:
    calculate the centroid of each cluster i
    set $c_i$ = to centroid i

Return sets c and r.  R is the cluster for each point in s, and c is the centroid of each cluster.
```

Table 1: The K-Means Algorithm

2.2 I-VT classifies eye movements by comparing them to a fixed velocity threshold [2]. Points which fall below that threshold are marked as being part of a fixation while points exceeding the threshold are classified as saccades. I-VT is largely accepted as the simplest form of eye movement classification. Nonetheless, few of more modern classification techniques substantially outperform I-VT.

2.3 Fixations and Saccades are two of the six major eye movement types which include: fixations, saccades, smooth pursuit, optokinetic reflex, vestibulo-ocular reflex, and vergence [2]. Fixations are defined as the focus of a subject being held on a single point. Saccades by contrast are the rapid movements between fixations. Although there are many sub-types of saccades this work focuses on the broad classification of fixations and saccades.

3. Method

This project applied a 3 step technique to the task of eye movement classification. These were:

1. Calculate the absolute velocity for each point as the geometric mean of the x and y velocities.
2. Calculate the acceleration of each point as the absolute difference of the velocities.
3. Filter outlier data points using a moving median.

4. Cluster the data using K-means.
5. Assign behavior labels based on the mean velocity of each cluster.

Each of these steps will be discussed in greater depth below.

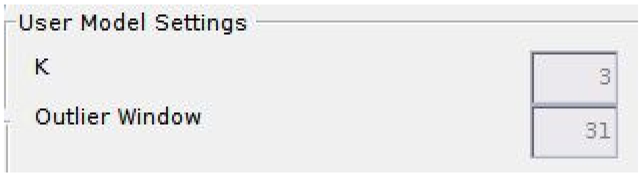
3.1 Moving Median is a method of outlier detection provided by Matlab. A median is calculated across a sliding window of user defined width. A new array is returned by the function which contains a 1 for every point which is more than 3 median absolute deviations away from the median, and a 0 for every other point. The code generated for this project marked every outlier detected by this function as Noise and set the velocity and acceleration of the point to 0. This is done to correct the skewing that outlier data points can produce in K-Means. A window size of 31 was used for all experiments in this project.

3.2 K-Means defines and then adjusts a user-defined number of centroids in data. The pseudo-code for this algorithm is presented in Section 2 of this paper. Clustering was performed on a 4 x n array containing the X-velocity, Y-velocity, absolute velocity, and acceleration for each data point. A K value of 3 was used for all experiments in this project. The K value needs to be the same as the number of behaviors being labeled and any other value will produce meaningless results. It is important to note here that K-Means is not a deterministic algorithm and that it occasionally will produce results that are drastically different from the average case.

3.3 Mean Cluster Velocity was calculated for every cluster after K-Means had been applied. This was calculated as the sum of all of the absolute velocities for each point in the cluster divided by the number of points in the cluster. Behaviors were then assigned based on the following assumption:

- Slowest Cluster -> Fixation
- Middle Cluster -> Pursuit
- Fastest Cluster -> Saccade

3.4 Parameters used in this project were k=3 and an outlier detection window of 31-points. The k value of 3 is necessary but the window size is variable. However, window sizes that are much larger (greater than 100) or much smaller (less than 10) will produce greatly degraded results while window sizes within that range are roughly equal.


User Model Settings

K	3
Outlier Window	31

Figure 1: screenshot of the parameters used.

4. Challenges

As with any novel technique this project experienced some challenges, some expected and some not. K-Means is sensitive to some kinds of noise. Specifically a few high-velocity or high-acceleration points could form a cluster of their own and push all of the remaining points into 1 or 2 clusters. This created cases were a single point was classified as a Saccade, a few others as Pursuits, and then the vast majority remaining as Fixations. This challenge was addressed by adding a denoising step to the approach but even that couldn't fully mitigate this danger.

Another challenge faced by this project was that K-Means is not fully deterministic. It is much more difficult to tune an algorithm when there is some variation between runs even with no changes having been made. This challenge is largely overcome by the fact that the parameters for this approach do not need to be tuned so once the code was finalized there were few further changes to be made. One lingering aspect of this challenge is that an occasional bad initialization will result in very poor clusters. This condition can cause every data point to be marked as noise or as a single behavior. This particular trouble cannot be mitigated without rewriting the implementation of K-Means used in Matlab.

Intuitively very negative and very positive velocities both indicate a Saccade since they mean that an eye is moving rapidly in one direction or its opposite. However, the euclidean distance measure still finds great dissimilarity between the two points described. This project accounted for this challenge by only considering the absolute values of the velocity and acceleration.

Finally, testing has shown that the cluster in eye tracking data are not as "crisp" as had been hoped going into this project. Intuitively there should be three groups in the data: high velocity, high acceleration saccades; mid velocity, low acceleration pursuits; and low velocity, low acceleration fixations. In reality though, these groups overlap substantially and tend to create one large blob rather than 3 easily discernible clusters.

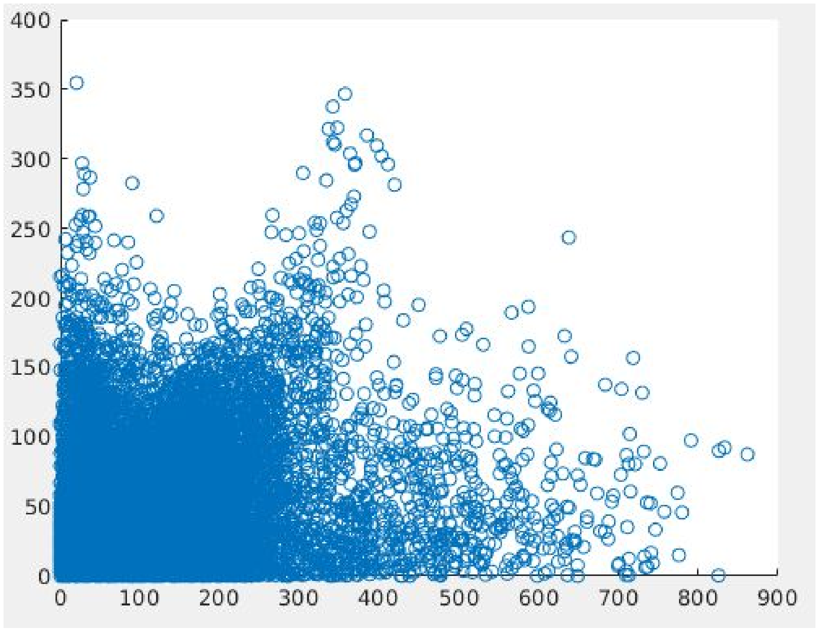


Figure 2: s001 charted with velocity on the horizontal axis and acceleration on the vertical.

5. Results

The eye tracking algorithm employed by this project produced the following classification:

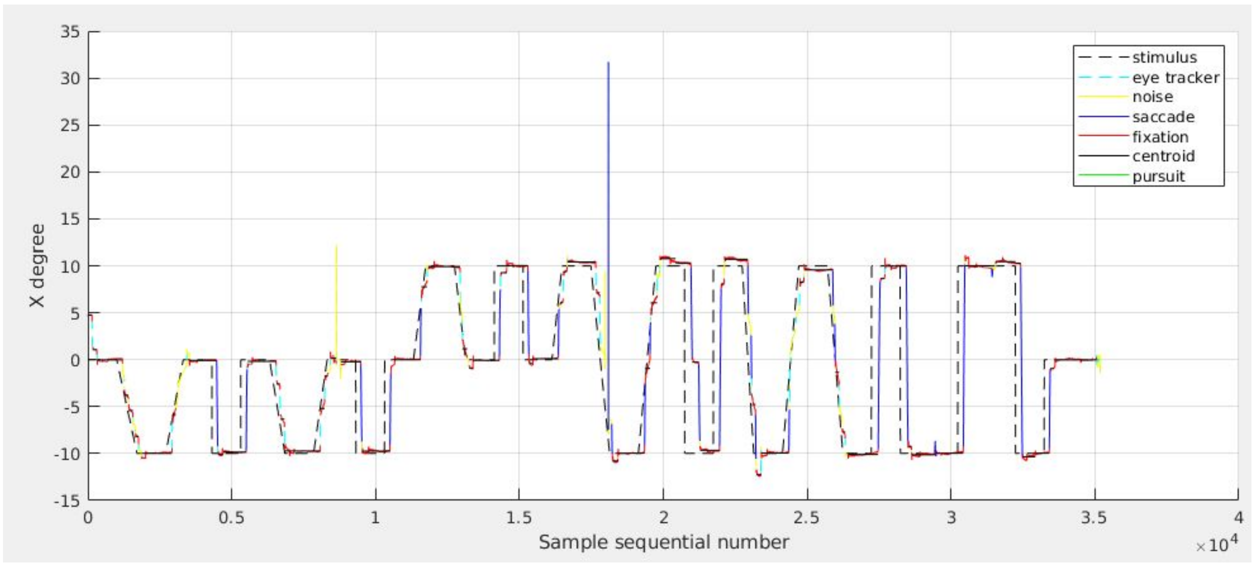
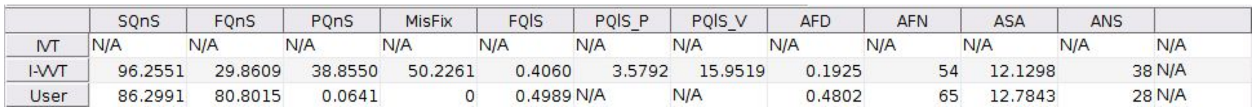


Figure 3: plot of labels eye tracking points in s007 with colored behaviors.

As can be seen most points were labeled as fixation, with some marked as saccades, and very few marked as smooth pursuits. Specifically 33,225 were labeled as Fixations, 1,346 were labeled as Saccades, 5 were labeled as Pursuits, and 661 were flagged as noise. This is in stark contrast to the results produced by I-VVT which mostly labeled the data as smooth pursuits divided by saccades. The scores rendered by the program are as follows:



	SQnS	FQnS	PQnS	MisFix	FQlS	PQlS_P	PQlS_V	AFD	AFN	ASA	ANS	
IVT	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
I-VVT	96.2551	29.8609	38.8550	50.2261	0.4060	3.5792	15.9519	0.1925	54	12.1298	38	N/A
User	86.2991	80.8015	0.0641	0	0.4989	N/A	N/A	0.4802	65	12.7843	28	N/A

Table 2: the scores generated by the Matlab code provided.

The scores reported by the provided program are as follows:

- Fixation Qualitative: 80.8
- Saccade Qualitative: 0.5
- Saccade Quantitative: 86.3
- Pursuit Quantitative: 0.1
- Pursuit Qualitative_P: n/a
- Pursuit Qualitative_V: n/a
- MixFix: 0

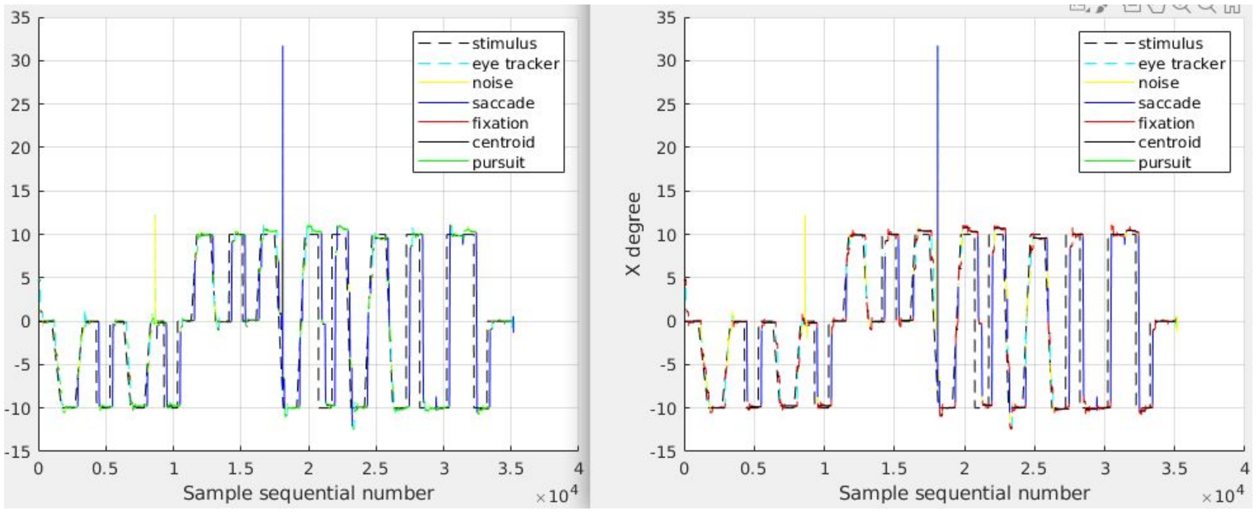


Figure 4: the results of I-VVT classification (left) alongside K-Means (right) on s007

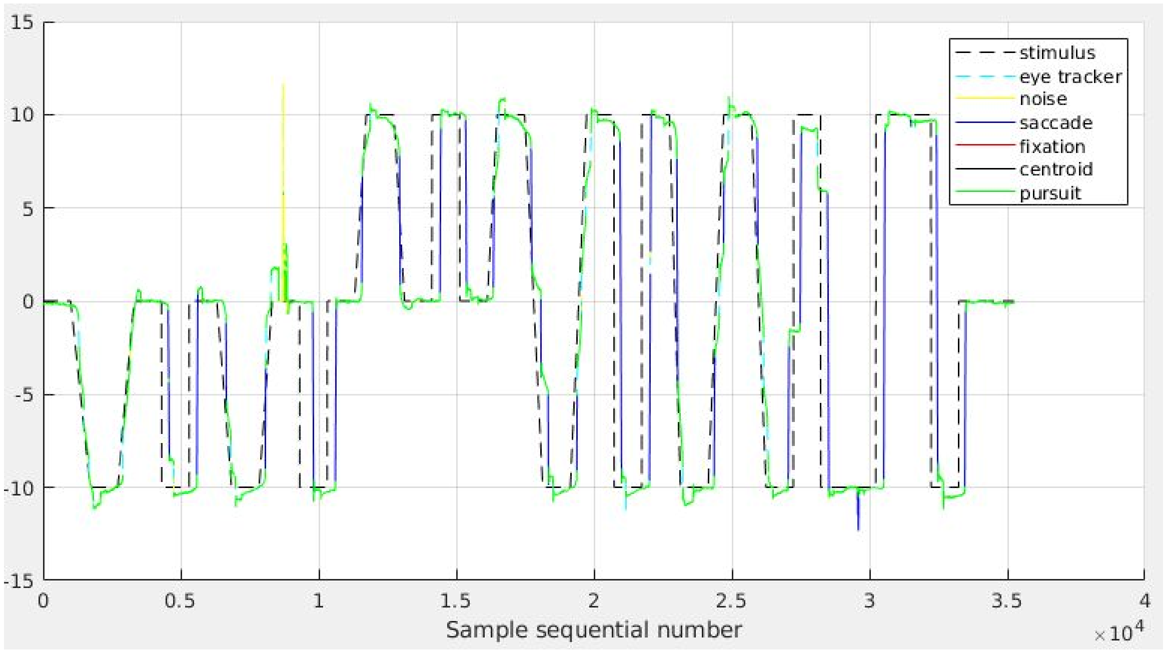
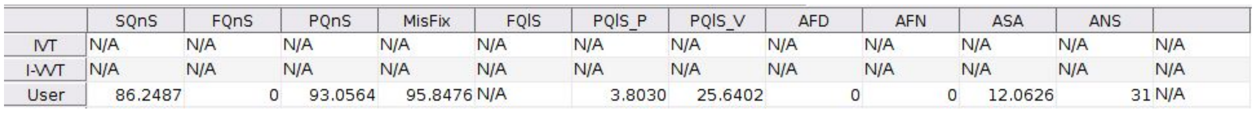


Figure 5: the results of K-Means run on s004. We can see in this image that the classifications produced are much more similar to what would be expected from I-VVT.



	SQnS	FQnS	PQnS	MisFix	FQlS	PQlS_P	PQlS_V	AFD	AFN	ASA	ANS	
IVT	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
I-VVT	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
User	86.2487	0	93.0564	95.8476	N/A	3.8030	25.6402	0	0	12.0626	31	N/A

Table 3: scores of K-Means on s004 data.

6. Discussion and Conclusion

Although K-Means did not surpass or even equal I-VVT in terms of accuracy when compared to ground truth, this does not mean that the results are meaningless and invalid. Exploratory techniques which reveal properties of data can be every bit as useful as analytical techniques that reliably detect understood principles. Furthermore clustering is a technique which is closely related to the practice of grouping data points in behavioral classifications. So exploring clustering (K-Means or otherwise) is an important approach to this problem even if it does not always prove to be the most accurate solution.

K-Means has demonstrated in previous work (Project 1) that it can accurately, quickly, and reliably distinguish between Fixations and Saccades. The results presented in this project demonstrate that this solution does not generalize easily. However, the work on datasets other than s007 in this project have shown that K-Means is still a viable approach for Pursuit detection on sufficiently clean data.

Ultimately better robustness in the presence of noisy data will be necessary for K-Means to be safely applied to the problem of Pursuit identification. It was often the case when working on set s007 that a few high-velocity or high-acceleration points would form a cluster of their own and push all remaining points into fixations. This remained true even after the data cleaning which was implemented in this project.

7. Future Work

Further work with pre-processing and de-noising the provided data will be the most efficacious means of improving the results produced by the products of this project. Some approach such as normalization or a more robust outlier detector may be what is needed to make K-Means a reliable approach to Pursuit classification.

An alternative (or additional approach) to removing noise from the data would be to work with clustering algorithms which are more robust to noise. Density based methods such as DBScan have demonstrated robust and impressive abilities in automatic noise detection. The risk is that density based methods do not produce a specified number of clusters. This means that the produced number would have to be merged or divided in order to produce the required set of 3 clusters. Possibly a hybrid approach could be applied wherein one clustering algorithm is applied for denoising and another for behavior classification. But that approach would be computationally expensive.

The problem may also be solved by expanding the number of features represented in the dataset. Data which appears noisy when only viewed as a pair of velocity and acceleration might prove to be tolerably regular if further dimensions are taken into consideration.

[1] McQueen J., (1967) Some methods for classification and analysis of multivariate observations, Proceedings of the 5th Berkeley Symposium on Math, Statistics, and Probability. Berkeley, CA

[2] Komogortsev O., Koh D., Jayarathna S., Gowda S., (2009) Qualitative and Quantitative Scoring and Evaluations of the Eye Movement Classification Algorithms, Technical Report: Texas State University. San Marcos, TX