Course project

Main target of this project was to analyze and visualize eye-tracking data. To achieve that I wrote a java program:

Compile:

javac FixationDetection.java

Run:

java FixationDetection trains.csv subject_ids+ [Options]

Options:

--show=subject_number - shows eye path of subject with fixations.
--dt=? default: --dt=70 - sets dispersion threshold
--min=? default: --min=100 - sets minimum fixation duration
--s=? default: --s=90 - sets minimum saccade amplitude
--tolerance=? default: --tolerance=3 – sets noise tolerance

Examples:

java FixationDetection trains.csv s8 s18 s28 s4 s14 s24
default use for group 8

java FixationDetection trains.csv s8 s18 s28 s4 s14 s24 --dt=50 --min=100
sets dispersion threshold=50 and minimum duration=100

java FixationDetection trains.csv s8 --show=0
shows visualization of first "s8" sample

Outputs:

The program always produces 3 files:

means.csv – statistics csv file with following structure: subject_id MFD_true MFD_SD_true MFD_false MFD_SD_false MSA_true MSA_SD_true MSA_false MSA_SD_false MFD_overall MFD_overall_SD MSA_overall MSA_overall_SD

fixations.csv – list of all fixations with following structure: subject_id,known,duration,x,y

saccades.csv – list of all saccades with following structure: subject_id,known,amplitude

The following files: means.csv, fixations.csv, saccades.csv were produced with following parameters:
java FixationDetection trains.csv s8 s18 s28 s4 s14 s24
dispersion threshold = 50 (50 pixels of dispersion allowed)
minimum duration=100 (100ms if sample rate is 1000 Hz)
tolerance = 3 (up to 2 samples can be omitted in a row after exceeding threshold)
saccade amplitude = 90 (minimum 90 pixels of saccade amplitude)

Algorithm:

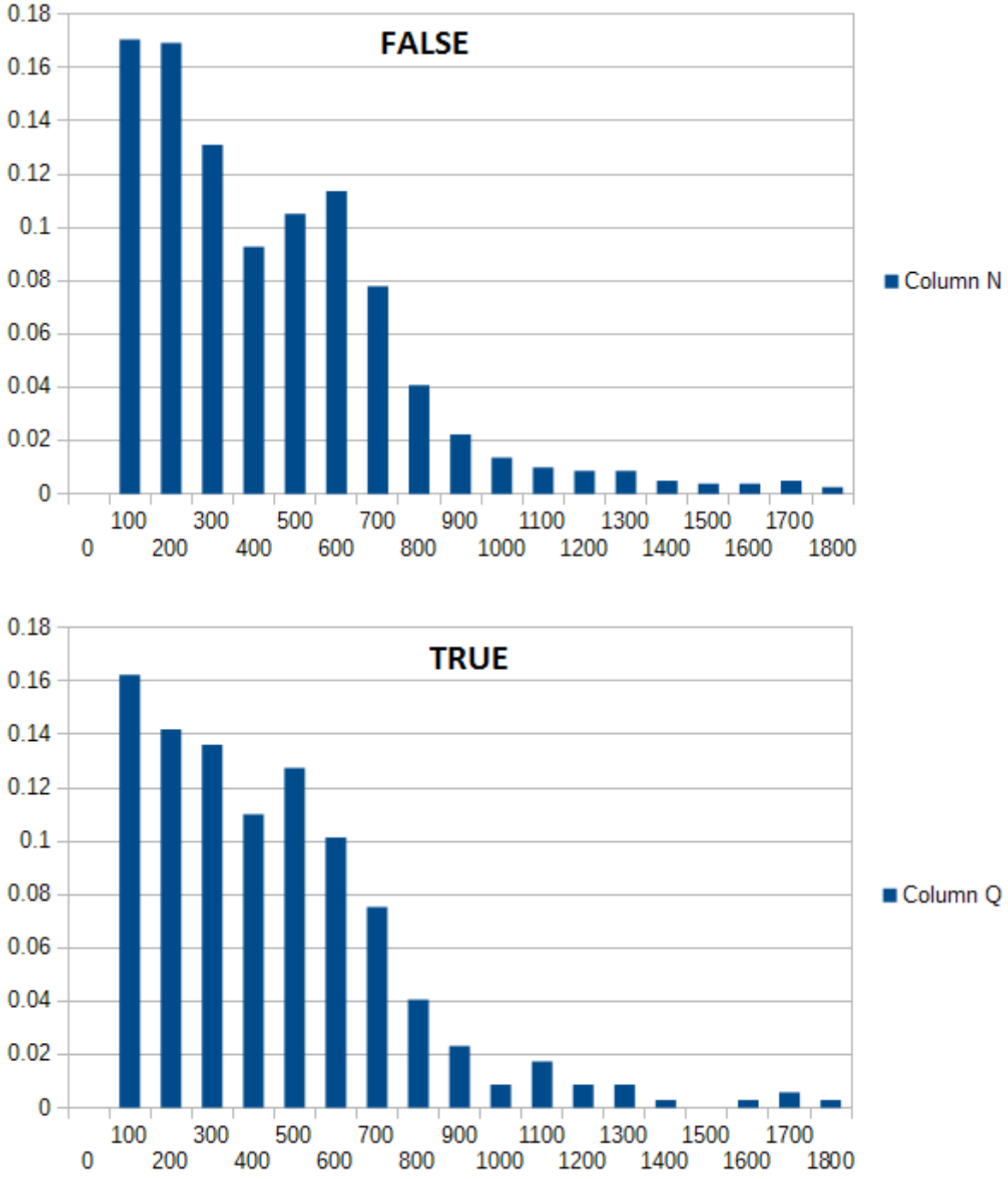
Saccade detection – velocity based algorithm
Fixation detection – dispersion threshold algorithm

```
ArrayList<Fixation> dispersionThresholdAlgorithm(String[] subject, double threshold, int
duration, int tolerance_threshold) {
     boolean reset = true;
     int count = 0;
     double minX = 0;
     double maxX = 0;
     double minY = 0;
     double maxY = 0;
     double oldMinX = 0;
     double oldMaxX = 0;
     double oldMinY = 0;
     double oldMaxY = 0;
     int tolerance = 0;
     ArrayList<Fixation> fixations = new ArrayList<>();
     for (int i = 4; i < subject.length; i = i + 2) {
          count++;
          double x = Double.parseDouble(subject[i]);
          double y = Double.parseDouble(subject[i + 1]);
          if (reset) {
               minX = x;
               maxX = x;
               minY = y;
               maxY = y;
               oldMinX = x;
               oldMaxX = x;
               oldMinY = y;
               oldMaxY = y;
               reset = false;
          }
          if (x < minX) {
               oldMinX = minX;
               minX = x;
          }
          if (x > maxX) {
               oldMaxX = maxX;
               maxX = x;
          }
          if (y < minY) {
               oldMinY = minY;
               minY = y;
          }
          if (y > maxY) {
               oldMaxY = maxY;
               maxY = y;
          }
          double dispersion = Math.max(maxX - minX, maxY - minY); //very simplified
          if (dispersion > threshold) {
               tolerance++;
               minX = oldMinX;
               maxX = oldMaxX;
               minY = oldMinY;
               maxY = oldMaxY;
               if (count >= duration && tolerance >= tolerance_threshold) {
                    Point2D center = new Point();
                    center.setLocation((maxX + minX) / 2, (maxY + minY) / 2);
                    fixations.add(new Fixation(center, count, (maxX - minX > maxY - minY ?
maxX - minX : maxY - minY)));
                    count = 0;
```

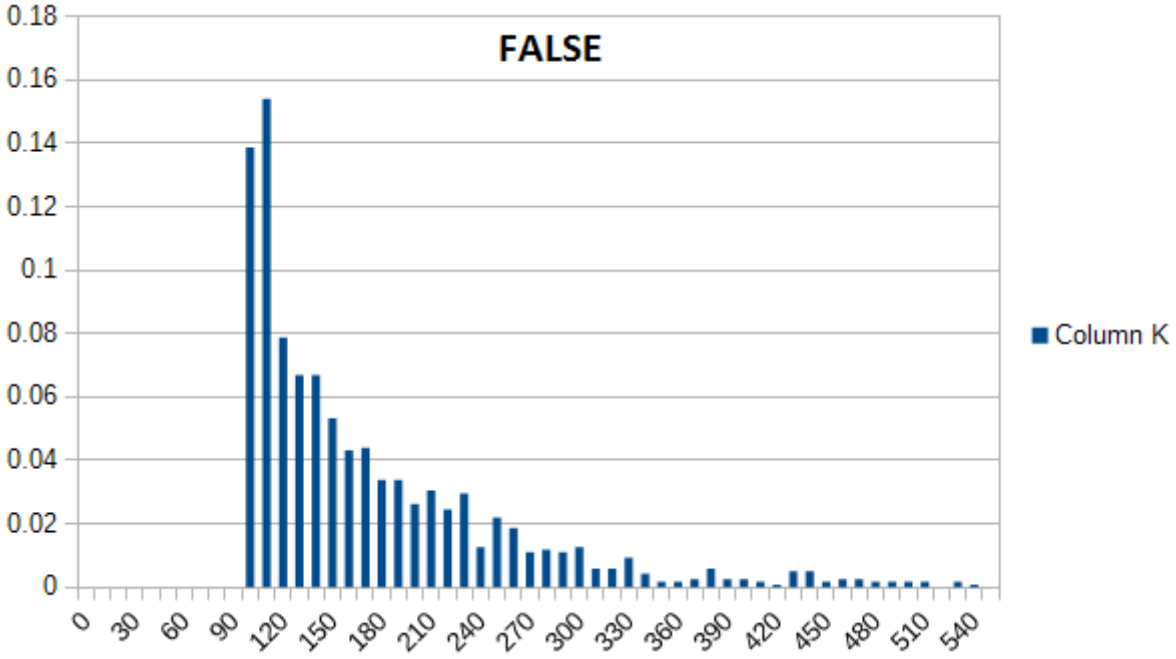
```
                reset = true;
            } else if (tolerance >= tolerance_threshold) {
                i = i - (count * 2) + 2;
                count = 0;
                reset = true;
            }
        } else {
            tolerance = 0;
        }
    }
    if (count >= duration) {
        Point2D center = new Point();
        center.setLocation((maxX + minX) / 2, (maxY + minY) / 2);
        fixations.add(new Fixation(center, count, (maxX - minX > maxY - minY ? maxX -
minX : maxY - minY)));
    }
    return fixations;
}


ArrayList<Saccade> saccadeDetection(String[] subject, double threshold, double
amplitude_threshold, int tolerance_threshold) {
    int count = 0;
    double realDistance = 0;
    Point2D start = new Point();
    Point2D current = new Point();
    Point2D prev = new Point();
    start.setLocation(Double.parseDouble(subject[2]), Double.parseDouble(subject[3]));
    int tolerance = 0;
    ArrayList<Saccade> saccades = new ArrayList<>();
    for (int i = 4; i < subject.length; i = i + 2) {
        count++;
        current.setLocation(Double.parseDouble(subject[i]), Double.parseDouble(subject[i +
1]));
        prev.setLocation(Double.parseDouble(subject[i - 2]), Double.parseDouble(subject[i -
1]));
        double velocity = current.distance(prev);
        double amplitude = current.distance(start);
        if (velocity < threshold) {
            tolerance++;
            if (amplitude >= amplitude_threshold && tolerance >= tolerance_threshold) {
                saccades.add(new Saccade(start, current, count, realDistance));
                count = 0;
                start.setLocation(current);
                realDistance = 0;
            } else if (tolerance >= tolerance_threshold) {
                count = 0;
                start.setLocation(current);
                realDistance = 0;
            }
        } else {
            tolerance = 0;
            realDistance += velocity;
        }
    }
    if (current.distance(start) >= amplitude_threshold) {
        saccades.add(new Saccade(start, current, count, realDistance));
    }
    return saccades;
}
```

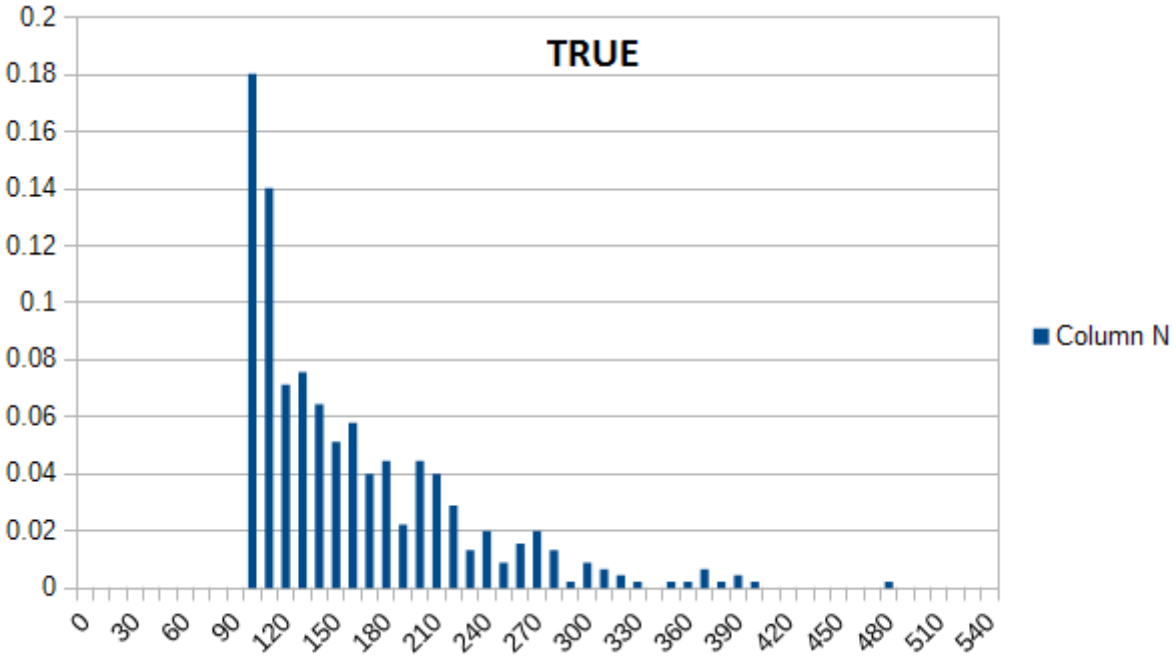
Results:

Saccade amplitude for known=true and known=false

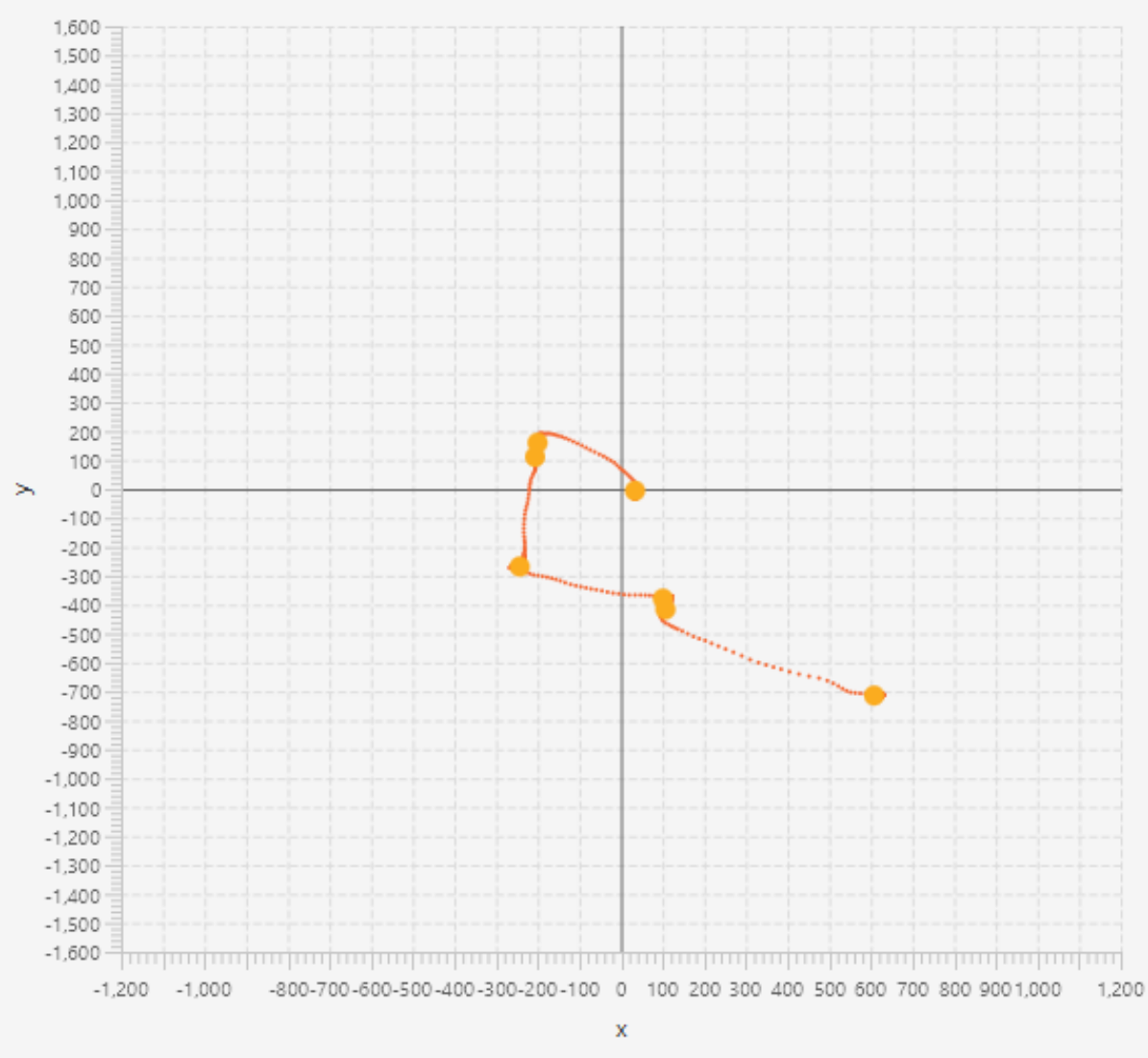


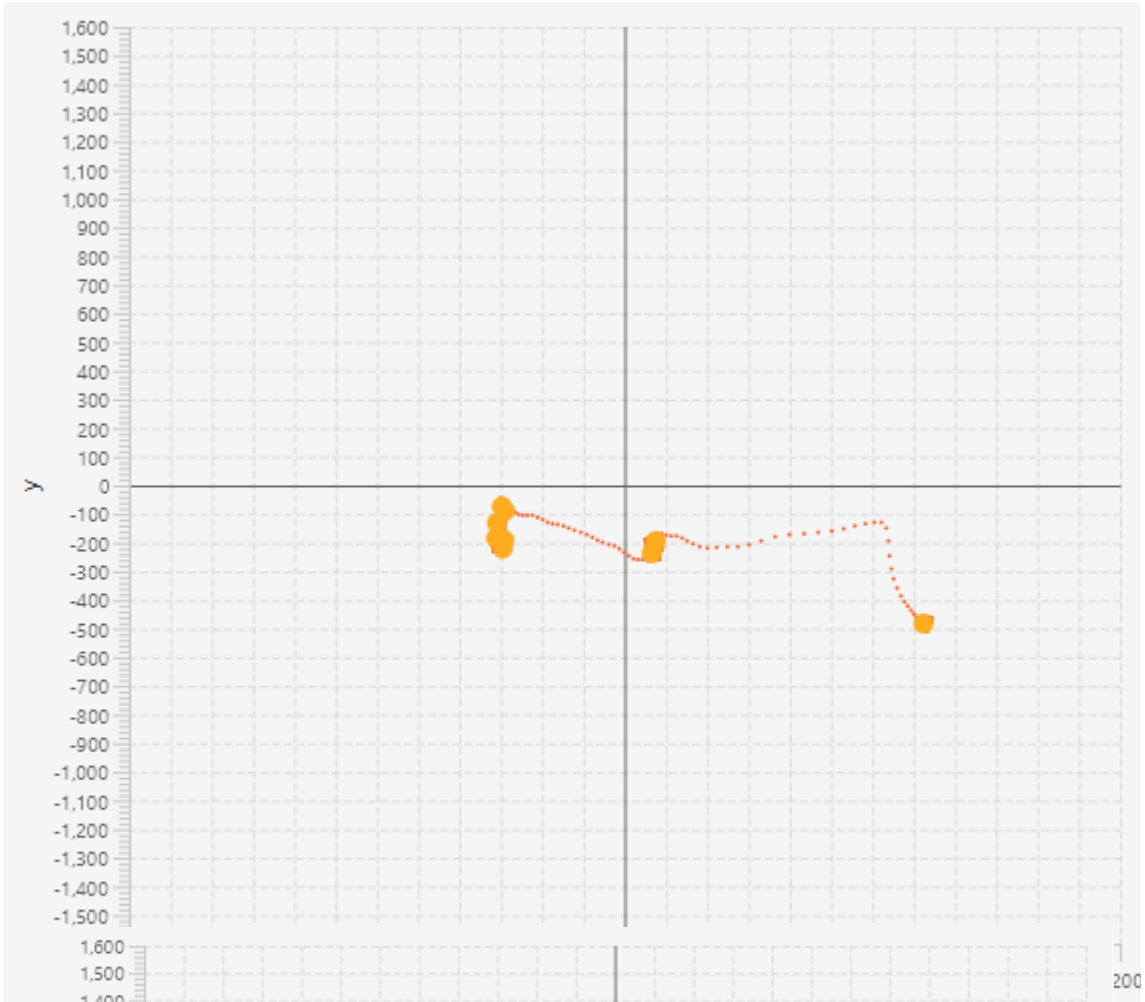
Fixation duration for known=true and known=false:

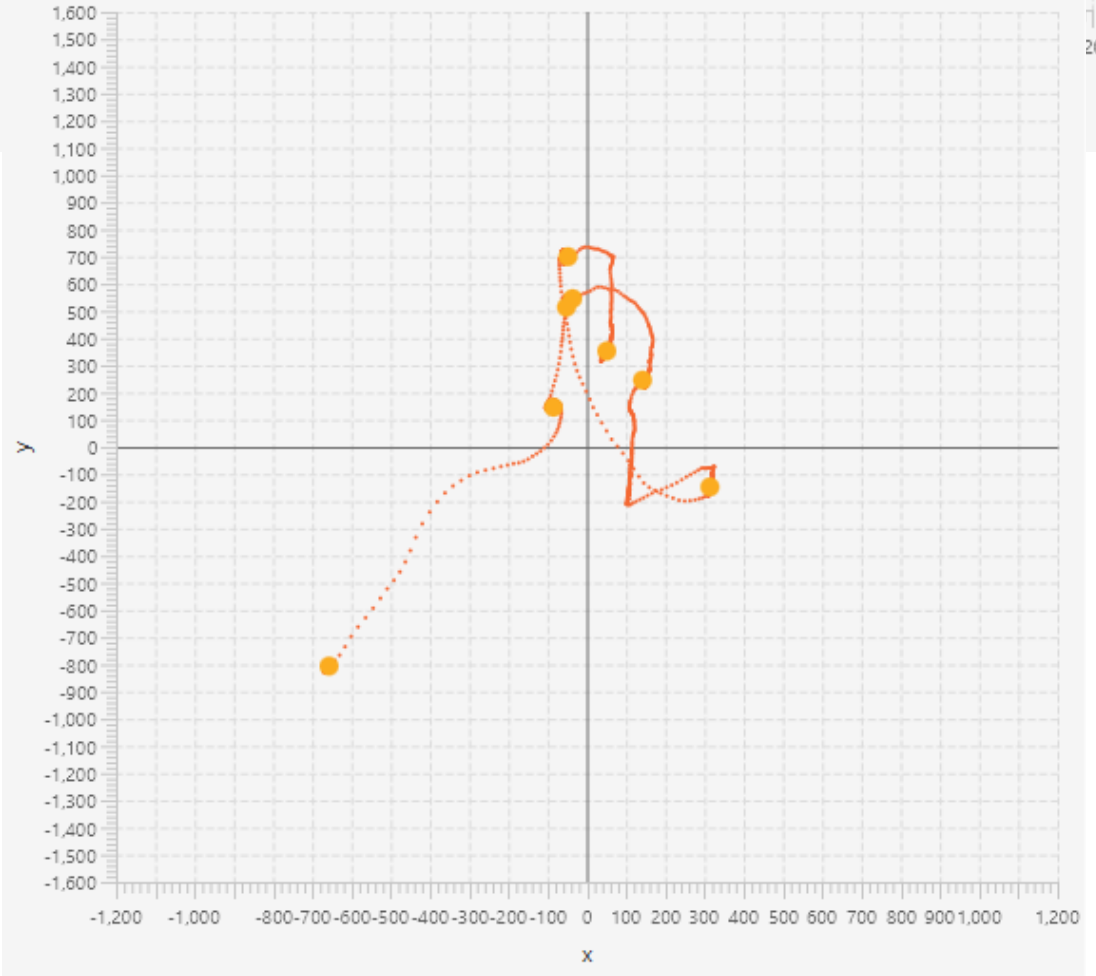




Some visualization of fixations with eye-path




1,600
1,500
1,400
1,300
1,200
1,100
1,000
900
800
700
600
500
400
300
200
100
0
-100
-200
-300
-400
-500
-600
-700
-800
-900
-1,000
-1,100
-1,200
-1,300
-1,400
-1,500
y


1,600
1,500
1,400
1,300
1,200
1,100
1,000
900
800
700
600
500
400
300
200
100
0
-100
-200
-300
-400
-500
-600
-700
-800
-900
-1,000
-1,100
-1,200
-1,300
-1,400
-1,500
-1,600
y
-1,200
-1,000
-800
-700
-600
-500
-400
-300
-200
-100
0
100
200
300
400
500
600
700
800
900
1,000
1,200
x