Experimental framework for the quantification of the dynamic visual acuity

Roberto Cardona[1], Jean-Christophe Fillion-Robin[2]
School of Computer and Communication Sciences (I&C)
Ecole Polytechnique Fédérale de Lausanne
CH-1015 Lausanne, Switzerland

***Abstract* — Using data collected from people by using a treadmill, an LCD screen and controlling software, we propose a framework to compute, in real time, the static and visual dynamic acuity. Working in collaboration with the medical doctors and vision specialists Prof. Marco Pellizzone and Prof. Jean-Phillipe Guyot from the University Hospital of Geneva, and combing our expertise in software design and data analysis, we were able to design and implement a robust framework.**

This paper highlights and contextualizes our implementation and outlines the crux of the problem in order to motivate future related research.

I. Introduction & Motivation

Over the period of eight months we got closer to answering the question about the quantification of dynamic visual acuity. While at it, a wide variety of different topics had to be addressed: ranging from ophthalmological aspects, psychometric function fitting, human response adaptive algorithms to user interface usability and software engineering.

The project was built on top of two cornerstones: the research carried out and experimental results obtained by students in previous semesters as well as the multiple enriching discussions with specialists from diverse areas at the University Hospital of Geneva.

By far and above the greatest obstacle was being able to comply with the experimental protocol that was dictated by the medical specialists. All the different components of the experimental framework had to be thoroughly validated against it in order to ensure that the collected data was useful and hence the resulting conclusions would lead to meaningful results. While providing value to the project, it was made clear to the team from the beginning, that the programmatic implementation of the experiment was secondary: its validity depended exclusively on the compliance with the protocol defined by the lead scientists.

The most important component of this framework was the developed software that served not only as guide during the experiments but also provided data acquisition and pre/postprocessing functionalities. An important aspect that must be mentioned is that the application had to minimize the risk of operator error to guarantee data accuracy, something that is non-trivial when designing software interfaces.

Experimental results were gathered which led to some conclusions from these. However, a larger sample is required to be able to draw more meaningful information. There is still a large amount of work to be done and, with this document in hand, any master-level engineering student should be able to take this project further. Despite of what is outstanding, the overall satisfaction at this point in time from both, team members and collaborators, is high.

The remaining of this document aims at providing a theoretical background for the topic in question, illustrating the different decisions made as well as describing project phases.

II. The Vestibular System

In order to maintain balance, the human body has an apparatus called vestibular system that takes care of providing the adequate information to both the visual system as well as the muscles that maintain the body stabilized. The main components of this system are depicted in Figure 1.

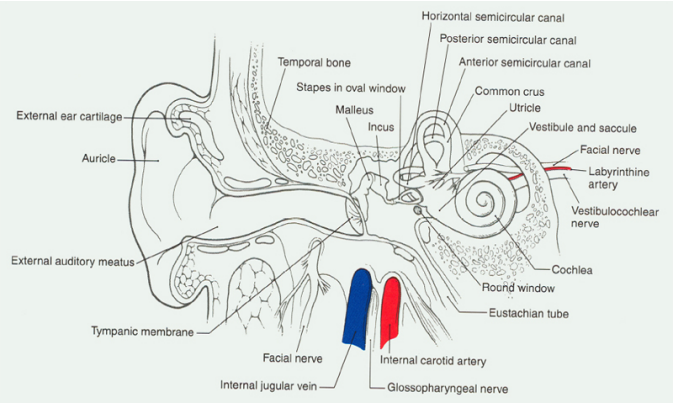


Figure 1: Anatomy of the human vestibular system [1]

While in movement, this same system is responsible for adjusting information (see Figure 2) received from other systems and passing it to the visual system so that it is able to take these into consideration when passing images about objects in sight to the brain. Patients with illnesses affecting this system have issues to correctly recognize text and numbers

Report written June 12, 2007.

1. e-mail: roberto.cardona@epfl.ch
2. e-mail: jean-christophe.fillion-robin@epfl.ch

while in motion. These are the target patients that will be objects of experiment with the developed framework.

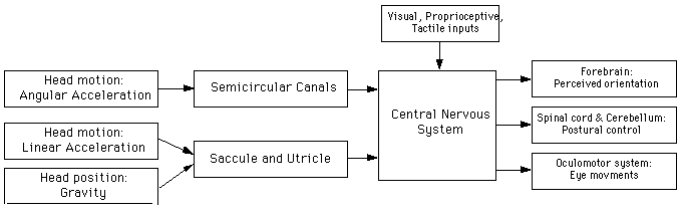


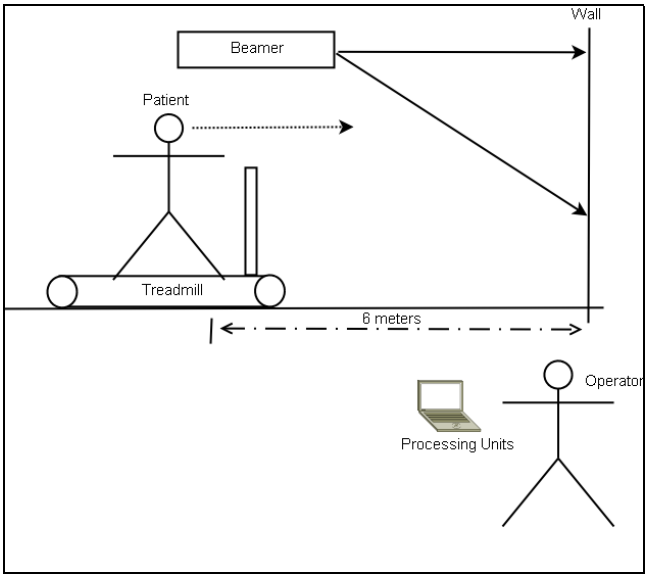
Figure 2: Functional diagram of Vestibular system [2]

III. Experimental Setup

As described in section I, the main purposes of the research in question is the acquisition of data relative to the reading abilities of human beings in motion. We looked for an easily-reproducible and controllable way of making the person runs or walks while trying to read. Two main options were presenting to us, either add exogenous (using a moving or vibrating chair) or endogenous (using a treadmill) motion. The latter one presented interesting characteristics. By using a treadmill, we are able to modulate the motion factor with ease. Moreover, the analysis of the results is simplified since the data acquired can be expressed as a function of the speed without major difficulties.

A. Initial setup

The initial implementation consisted of a display device, a treadmill, an audio acquisition instrument, a head tracker [8] and a central processing unit (personal computer) allowing the operator to control experiment's evolution (see Figure 3).



1)Beamer

After initial testing, the beamer revealed limitations in terms of resolution. At a reading distance of six meters, the smallest symbol to be displayed is approximatively five millimeters wide (see Table 1). The granularity of the beamer pixel grid was 2 millimeters which forced us to consider other display alternatives.

V.A	LogMAR	Min. sep *Arcmin*	Thick. (mm)	Width (mm)
0.100	-1	10	17.45330	87.27
0.125	-0.9	8	13.96264	69.81
0.167	-0.78	6	10.47198	52.36
0.200	-0.7	5	8.72665	43.63
0.250	-0.6	4	6.98132	34.91
0.333	-0.48	3	5.23599	26.18
0.400	-0.4	2.5	4.36332	21.82
0.500	-0.3	2	3.49066	17.45
0.667	-0.18	1.5	2.61799	13.09
0.800	-0.1	1.25	2.18166	10.91
1.000	**0**	**1**	**1.74533**	**8.73**
1.250	0.1	0.8	1.39626	6.98
1.667	0.22	0.6	1.04720	5.24
2.000	0.3	0.5	0.87266	4.36

2) Audio acquisition

The initial purpose of the audio acquisition was the real time analysis of the patient answers and the measurement of the reaction time. Real time voice analysis is a interesting way of minimizing the probability of error introduced by the external operator. However, this last method isn't error free and need a fine tuning.

3) Head tracker

The head tracker would allow for obtaining head position and movement information while the patient is walking or running on the treadmill. The hypothesis behind the head motion analysis concerned the observation of a potential correction or compensation of the head position when the patient would have proven some difficulties to read the symbols. The third part software provided with the head tracker gives a simple and easy way to gather relative three-dimensional position of the head and its acceleration

4)Symbols

Initially, each experiment was composed of two sub-tasks, a task to compute the static visual acuity where the patient has to determine the apparent orientation of the Landolt-C character [10], and a task to compute the dynamic visual acuity where the patient has to read a number randomly picked between zero and nine.

B.Current setup

Since the beamer doesn't have a enough resolution, the use of a high resolution LCD screen is a possible alternative. Using a flat screen with a native resolution of 1280x800 pixels and a diagonal length of nineteen inches is an adequate solution (see Table 2). The number of pixels used to display the smallest symbols is large enough to allow the patient to clearly recognize these (see Figure 4).

The current setup doesn't include neither the head tracker nor the audio acquisition device. This in favor of reducing the complexity of the data and giving more attention to the compliance with the experimental protocol.

Resolution	*640*	*480*	*800*	*600*
Aspect ratio	1.33		1.33	
Screen size (mm)	386.08	289.56	386.08	289.56
Pixel size (mm)	0.6	0.6	0.48	0.48
Char. Size / v.a.:0.1 (#pixels)	7.23	7.23	9.04	9.04
Char. Size / v.a.:2.0 (#pixels)	144.66	144.66	180.83	180.83
	1024	*768*	*1280*	*800*
	1.33		1.6	
	386.08	289.56	409.24	255.78
	0.38	0.38	0.32	0.32
	11.57	11.57	13.65	13.65
	231.46	231.46	272.95	272.95

Table 2: Number of pixel required as function of screen resolution and diagonal length

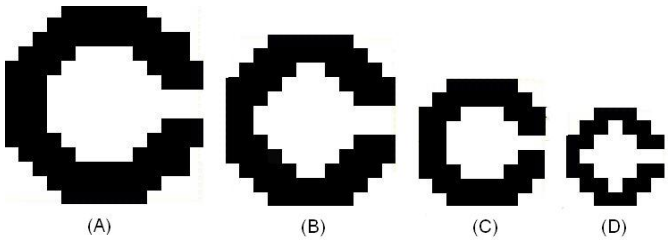


*Figure 4: Shape of the Landolt-C displayed with a 19inches screen at resolution 1280*800 (A), 1024*768 (B), 800*600(C) and 640*480 (D)*

Both static and dynamic tasks have been merged, each time that the operator wants to start a new experiment, he/she will have to type the patient data, then enter the corresponding treadmill speed and select the side corresponding to the tested eye. To facilitate the patient work, we switched to a task using only the Snellen optotypes [11]. Since these look like well-known letters and their efficiency to check visual acuity has been proved by optometrists, the patient won't have to think or interpret the result and, therefore, the visual acuity results won't be biased. The initial task dealing with the orientation of the Landolt-C was too demanding in terms of thinking.

IV. Adaptive Method

The algorithm used to adapt the size of each character displayed for the patient based on previous response is called Staircase or "2 down 1 up". It is a corollary to a series of research papers published in the field as well as practical experiences done at the University Hospital of Geneva with other type of stimuli and experimental setups [4] [5].

Some initial conditions: there are a maximum and minimum values on which the algorithm can operate. For this specific case, these bounds are the limits on the Snellen Chart. Also, a minimal and maximal step size must be defined which, for this case, the smallest is ¼ of the original step size which corresponds to four steps on the Snellen Chart.

The algorithm starts with the largest possible stimuli (in this case the character corresponding to visual acuity of 2.0 on the Snellen Chart) and it must get two consecutive positive responses from the patient to continue, otherwise it terminates.

Once this step is passed, the stimuli is reduced to by the initial step size (four at this point).

If a correct answer is given, the algorithm provides the same stimuli size again and, if correct, it goes down (reduces the stimuli size). This time, only half the number of steps than last time. This stage is called a peak. Otherwise, while in the middle of a peak, it goes up again by the same number of steps than the last transition.

If a negative answer is given, the stimuli is the next up by the current number of steps.

This procedure goes on again until the step size is reduced to its minimum and six valleys (the opposite of a valley) and six peaks. Then the algorithm converges. The average of these 12 values are taken to compute an average which is the visual acuity value for the patient.

Initially the number of valleys for convergence was four but, due to some bogus in the initial testing, six was used instead.

This algorithm typically converges in 30 to 40 iterations which is the point at which one obtains about 70% correct answers. It is important to mention that the value obtained doesn't correspond to a valid value on the Snellen chart [11] therefore it should only serve as some kind of reference on where the actual VA value of the patient lies. There are some interpolated values that can be used to obtain a higher level of accuracy but these were not incorporated.

Please refer to Figure 7 which illustrates the algorithm better.

V. Experimental Results

The initial hypothesis that the doctors presented was not validated, which stated that the dynamic visual acuity should remain around 1.0 for young healthy students under running conditions. However, something that can be drawn from the results is that the visual acuity is diminished when the patient is moving, either running or walking.

The details on the different individuals from the control group are attached to this report as an appendix.

	Individual 1		Individual 2		Individual 3		Individual 4		Individual 5	
	R	L	R	L	R	L	R	L	R	L
0 to 5 km/h	0.43	0.22	0.03	0.03	-0.1	-0.03	-0.12	0.39	0.33	0.33
5 to 9 km/h	0.13	0.23	-0.03	0.01	0.06	0.35	0.13	-0.08	-0.01	-0.03

Table 3: Decrease of Visual Acuity as a function of treadmill speed

Table 3 shows the decreases in different patients when the treadmill speed was increased. Positive values correspond to acuity increases. The mean of these values is 0.15 for the first row and 0.08 for the second row. Due to the ample variability in these results it is difficult to suggest a clear trend or pattern but it is possible to safely conclude that the acuity is overall decreased. The statistical properties give some hints on the patterns of the data but it is considered that these provide very limited information.

In order to make stronger conclusions one must test more patients as well as introduce more treadmill speeds to determine specific thresholds on when is the acuity diminishing. The framework in question should be fundamental in arriving at these conclusions.

The clear differences between various healthy adults in dynamic visual acuity is another important conclusion that validates one of the side hypothesis made by the medical specialists.

VI. Controller Software

The developed application (controller) allows to modulate, using the adaptive method presented in section IV, the size of the symbols displayed according to the patient answers. The patient answer aren't directly interpreted by the controller. An individual operating the computer has to input these answers.

Since the specification changed, the program interface evolved to match the new requirements. To minimize the future adaptation, the controller has been designed with enough flexibility.

The controller software runs on a dedicated computer; in our case a laptop was used. The LCD screen is then connected to the laptop in order to input answers and display symbols on the flat screen. The virtual desktop view by the operator is extended to the flat screen. That way, the controller software can easily display symbols without interfering with the operator's interaction. All low level tasks concerning the resolution adaptation and the management of screen parameters are delegated to the operating system and the video card driver.

To ensure a correct display of the symbols, a calibration mechanism has been included in the software. A simple black square having dimension corresponding to a visual acuity of one (8.7 mm width) is presented on the screen, the operator will have to check, using a simple ruler, if the square measures exactly the desired size. If not, an on-screen slider allows to match the desired value and determines the scaling factor. Later, this same scaling factor will be used to display and adjust symbols dimension.

The first draft of the application didn't include a specific button corresponding to patient's answers. The interpretation of the answer was done by the operator using the left mouse button for a correct answer and respectively the right one for wrong answer. (see Figure 5)

After discussion with the medical specialists, it was decided to

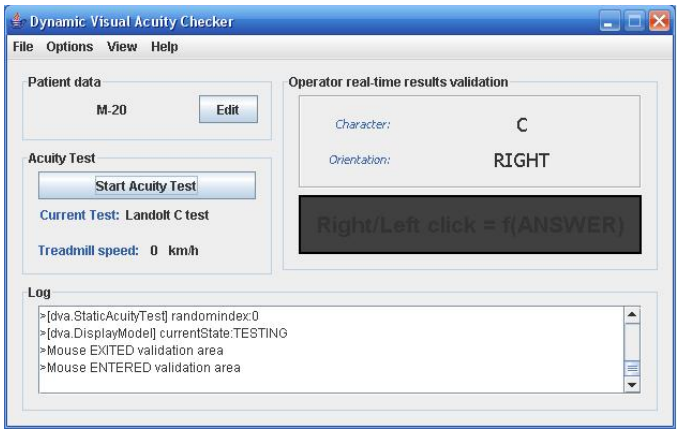


Figure 5: Initial interface of the controller software

change the initial design (see Figure 6). The patient's answers are now recorded and the operator doesn't have to interpret the answer. This enhancement decreases the operator's amount of work thus error probability. For example, if the patient's answer is 'C' and the real symbol is a 'S', the software will decide if the answer is correct or not.

The processing of the data is also an important issue. All results are recorded and future re-use of the data is facilitated by the creation of XML and Matlab files.

Using a third-party library [9], a chart corresponding to the patient's performance is generated. (see Figure 7)

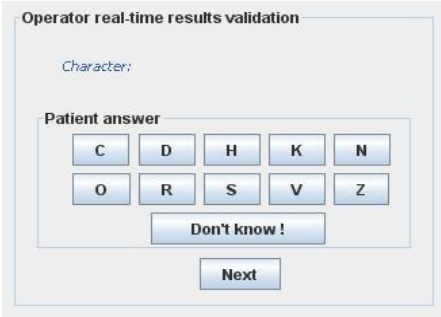


Figure 6: Redesigned interface with new operator input area

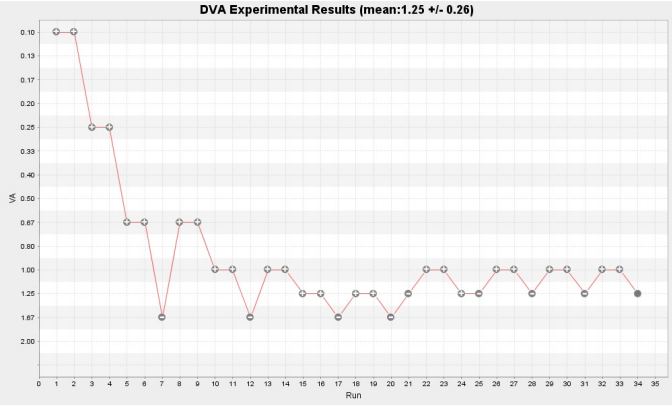


Figure 7: Example of chart generated by the controller - Woman, 21, glasses, left eye, 0 km/h

VII. Psychometric Function

In order to model the relationship between a parameter of the stimuli, in this case the size of the character, and the responses given by the patient, a psychometric function is used. [7]

In order to obtain this, one places all positive answers on the upper part of the graph, the incorrect ones on the lower and the transition between the two, which resembles a sigmoid, the psychometric function is fit. This is illustrated in Figure 8.

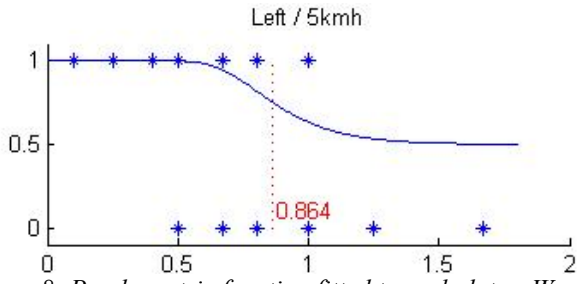


Figure 8: Psychometric function fitted to real data - Woman, 21, glasses, left eye, 5 km/h

In order to validate the results obtained for each patient, the psychometric function was fitted into their responses according to the procedure described before. The value in question lies in the sensory threshold and is exactly within the inflection point of the function.

This psychometric function is the fundamental underlying principle that support the Snellen Chart procedure used by optometrists. Due to imperfections in the experiments it is not possible to obtain the precise value but knowing where the sensory threshold is located is enough.

Psychometric functions were fitted using the Matlab toolbox *psignifit* version 2.5.6, a software package which implements the maximum-likelihood method described by Wichmann and Hill [3].

VIII. Outstanding Work

By far, the most important aspect that has to be taken care of is the validation of the tool with actual patients that suffer from problems with their vestibular system. Despite the efforts made by collaborators and team members, it was not possible to coordinate this during the established timeframe. However, with the data collected from healthy students and the results obtained, it is believed that the implementation is correct.

As mentioned before and due to time constraints, it was not possible to integrate the head-tracker system with the application nor into the experimental framework.

Additional improvements have to be made to the software in terms of operator error resiliency. One of the weaknesses discovered during the experiments was how easy it is for the operator to get out of synchronization with the patient's responses during the experiment. This can be solved by implementing a timer on the keypad which freezes it for some milliseconds after each character is displayed.

An additional feature that would reduce the number of errors in these situations is having the application generate a beep after each new character is displayed. This is especially useful in situations where the adaptive algorithm determines that two identical letters have to be displayed in a row with or without size difference. Also, when similar letters are displayed at a small size or one that is below the patient's DVA threshold, he/she may think that the letter has not changed and will not say anything thus confusing the operator.

It is also important to mention a feature that was initially built into the application: latin squares experiment scheduling method. This aims at avoiding correlation between experimental conditions and the way the experiments are scheduled for a given patient. In this context it is specific to the order in which the running, walking and static tests are scheduled for every patient. This feature was asked to be removed while the validity of the tool was confirmed and should be integrated again before real data is collected.

IX. References

[1] David Dickman, "*Vestibular System Primer*", http://vestibular.wustl.edu/vestibular.html

[2] Department of Neurophysiology, University of Wisconsin – Madison, "*The inner ear: The vestibular apparatus*" http://www.neurophys.wisc.edu/h&b/textbook/chap-7.html

[3] Wichmann, F. A. & Hill, N. J. (2001a): "*The psychometric function: I. Fitting, sampling and goodness-of-fit*". Perception and Psychophysics 63(8), 1293-1313.

[4] Levitt, H. "*Transformed Up-Down Methods in Psychoacoustics*" The Journal of Acoustical Society of America. 1971.

[5] Schlauch, R. & Rose R. "*Two-, three- and four-interval forced-choice staircase procedures: Estimator bias and efficiency*" The Journal of Acoustical Society of America. 1990.

[6] Tian J., Shubayev I. & Demer J. "*Dynamic visual acuity during passive and self-generated transient head rotation in normal and unilaterally vestibulopathic humans*"

[7] Boex C., Baud L.,Consendai G., Sigrist A., Kos I. and Pelizzone M. "*Acoustic to Electric Pitch Comparisons in Cochlear Implant Subjects with Residual Hearing*" Journal of the Association for Research in Otolaryngology, February 2006, p 114

[8] Motion Xsens Technologies, *3DOF Orientation Tracker for human motion capturing,* http://www.xsens.com/

[9] Java chart library, http://www.jfree.org/jfreechart/

[10] Landolt-C, http://en.wikipedia.org/wiki/Landolt_C

[11] Snellen Chart, http://en.wikipedia.org/wiki/Snellen_chart

X. Appendix

Some relevant parts of the code and the experimental results gathered will be presented here. Developed under GPL License, source code, results and specification are freely available here: http://code.google.com/p/dvachecker/

A. Staircase algorithms

```
/*
 * Staircase.java
 *
 * Created on May 25, 2007, 12:17 AM
 *
 */

package dva.util;

import [...]

/**
 * Staircase algo
 *
 * @author Roberto
 * @author J-chris
 *
 * Instructions
 *
  1) Create instance of object
  2) Call whatSize method with right or wrong arg to obtain the next size of char to display
  3) This method will return either the new size, -1 if the algo has diverged, -2 if max runs
exceeded and 0 if it has converged
  4)  If the return value stays the same the same stimulus must be displayed
  5)  Call the getConvergenceValue to obtain the converged value
  6)  It is assumed that the history of right and wrong responses will be stored by another
object.
 */
public class Staircase {

    final static int NUMBER_PEAKS_VALLEYS_REQUIRED = 6;

    //state variables
    private float stepSize; //current step size
    private float prevStepSize; //previous run step size
    private float curVal; //current value/level/size
    private float prevVal; //previous run value/level/size
    private int runDir; //1 going up, -1 going down
    private int runNumber; //number of runs so far
    private int peakIdx;
    private double peaks[]; //vector with peaks
    private int valleyIdx;
    private double valleys[]; //vector with valleys
    private double Lvalleys[];
    private double Lpeaks[];
    private boolean peaker; //var to identify double peaks
    private float minSizeCnt;
    private boolean converged; //whether the algo has converged yet
    private double convergenceVal;
    private double convergenceValStdDev;
    private int seriesCnt;
    private double[] curValHistory;
    private boolean[] curValResponses;
    private int numCVals; //# of consecutive valleys at size=1
    private final float LIMIT_UP = 15;
    private final float LIMIT_DOWN = 1.0f;
    private final float MIN_STEPSIZE = 1.00f;
    private final int MAX_RUNS = 60;

    //chart output
    private XYSeries seriesLinear;

    //files
    File outputdir = null;
    String fileprefix = "";

    /**
     * Creates a new instance of Staircase
     */
    public Staircase() {
        peaks = new double[50];
        peakIdx = 0;
        valleys = new double[50];
        valleyIdx = 0;
        curValHistory = new double[50];
        curValResponses = new boolean[50];
        seriesLinear = new XYSeries("Values-Linear");
        numCVals = 0;
        convergenceVal = 0;
        convergenceValStdDev = 0;


    }

    public void initSize (float initSize, float initStepSize, File outputdir, String fileprefix) {
        runNumber = 1;
        runDir = 1;
        curVal = initSize;
        stepSize = initStepSize;
        seriesCnt = 1;
        //start logging curVal
        curValHistory[seriesCnt-1] = initSize;
        curValResponses[seriesCnt-1] = true;
        seriesLinear.add(seriesCnt,initSize);
        seriesCnt++;
        peaker = false;
        minSizeCnt = 0;

        //file management
        this.outputdir = outputdir;
        this.fileprefix = fileprefix;
    }


    public float whatSize(boolean answer) {
        //prepare for next round
        prevStepSize = stepSize;
        prevVal = curVal;

        if(runDir == -1) { //descending
            if(answer && !peaker) { //check again
                stepSize = prevStepSize;
                peaker = true;

            }//close if answer
            else if (answer && peaker) { //keep descending if checked again
                stepSize = prevStepSize;
                curVal = prevVal - stepSize;
                peaker = false; //potential double correct

            }//close else if
            else if (!answer) { //reverse direction, start climbing

                //A lower/upper bound condition avoid out of bound access
                if (curVal>0 && curVal<15){
                    valleys[valleyIdx] = ScreenMapper.getVA( (int)curVal ); //sScale[14-(int)curVal];
//store valley information
                    valleyIdx++;
                }

                stepSize = prevStepSize;
                curVal = prevVal + stepSize;
                runDir = 1;

                if (peaker) peaker = false; //in case of false double correct
            }//close elseif answer

        } else if (runDir == 1) { //climbing

            if(!answer && !peaker) { //keep climbing
```

```
            stepSize = prevStepSize;
            curVal = prevVal + stepSize;

        } else if (!answer && peaker) { //keep on climbing, false alarm for double correct
            stepSize = prevStepSize;
            curVal = prevVal + stepSize;
            peaker = false;

        } else if(answer && !peaker) { //check again for a positive response
            stepSize = prevStepSize;
            curVal = prevVal;
            peaker = true; //we are at a potential double peak

        } else if(answer && peaker) { //we are at a double peak, invert direction

            //A lower/upper bound condition avoid out of bound access
          if (curVal>0 && curVal<15){
              peaks[peakIdx] = ScreenMapper.getVA( (int)curVal ); //sScale[14-(int)curVal]; //log
the peak
              peakIdx++;
          }

          if(runNumber>1) stepSize = (prevStepSize/2 >= MIN_STEPSIZE)? prevStepSize/2 :
MIN_STEPSIZE;
          curVal = prevVal - stepSize;
          runDir = -1;  //direction inversionZ
          peaker = false; //reset this var
          if (stepSize == MIN_STEPSIZE) numCVals++;
        }
    }

    DvaLogger.info(Staircase.class, "Current value:" + curVal + ", va:" +
ScreenMapper.getVA( (int) curVal));

    runNumber = (peaker) ? runNumber : runNumber+1; //do not increase number of runs if
we are in potential double peak
    if(stepSize == MIN_STEPSIZE && !peaker) ++minSizeCnt;

    double oldCurVal = 0;

    //check for convergence
    if (numCVals == NUMBER_PEAKS_VALLEYS_REQUIRED ){  //do not consider
minSizeCnt for now
        //DvaLogger.debug(Staircase.class, "valleyIdx:" + valleyIdx + ", valleys:" +
ArrayUtils.toString(valleys) );
        //DvaLogger.debug(Staircase.class, "peakIdx:" + peakIdx + ", peaks:" +
ArrayUtils.toString(peaks) );

        // ---------- Compute Mean and Standard deviation ----------
        double sum = 0; // sum of all value
        double sumOfSquare = 0; //sum of the square of all value

        for (int i=1; i <= NUMBER_PEAKS_VALLEYS_REQUIRED; i++){

            sum += peaks[peakIdx - i];
            sum += valleys[valleyIdx - i];

            sumOfSquare += peaks[peakIdx - i] * peaks[peakIdx - i];
            sumOfSquare += valleys[valleyIdx - i] * valleys[valleyIdx - i];
        }

        convergenceVal = sum / (NUMBER_PEAKS_VALLEYS_REQUIRED * 2);
        convergenceValStdDev = Math.sqrt(
sumOfSquare/(NUMBER_PEAKS_VALLEYS_REQUIRED * 2) - convergenceVal *
convergenceVal );

        converged = true;
        oldCurVal = curVal;
        curVal = 0;
        DvaLogger.info(Staircase.class, "convergenceVal:" + convergenceVal + ", stddev:
"+convergenceValStdDev);

    }//close if convergence


    //check min limit has not been surpassed
    if (curVal < LIMIT_DOWN && (curVal != 0 || numCVals < 6) ) curVal = LIMIT_DOWN;
//check if lower bound has been surpassed

    //keep track of patient answer
    curValResponses[seriesCnt-1] = answer;

    //plotting
    if (!converged)  {
        seriesLinear.add(seriesCnt,curVal);
        curValHistory[seriesCnt-1] = ScreenMapper.getVA( (int)curVal );
    }
    else {
        seriesLinear.add(seriesCnt,oldCurVal);
        curValHistory[seriesCnt] = curVal;

    }

    seriesCnt++;

    //various checks
    if (runNumber > MAX_RUNS) curVal = -2; //check if max number of runs exceeded
    if (curVal > LIMIT_UP) curVal = -1; //check if upper bound has been surpassed

    return curVal;
  }//close whatSize

 [...]

  public void doGraph(String param) throws ChartFileCreationException {
     [...]
  }

}
```

B.Results

All the raw data are available as XML and Matlab files here: http://dvachecker.googlecode.com/svn/trunk/results/ .

For each of the person of the control group, we add the chart generated by the controller software and the corresponding psychometric function.

1)Individual 1 - W, 22, glasses

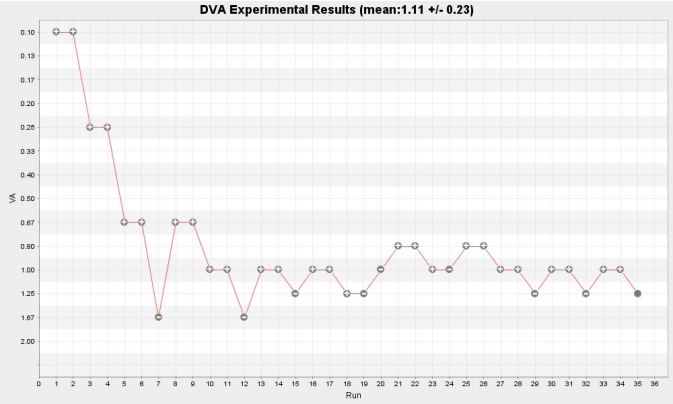


Figure 9: Left eye, 0 km/h

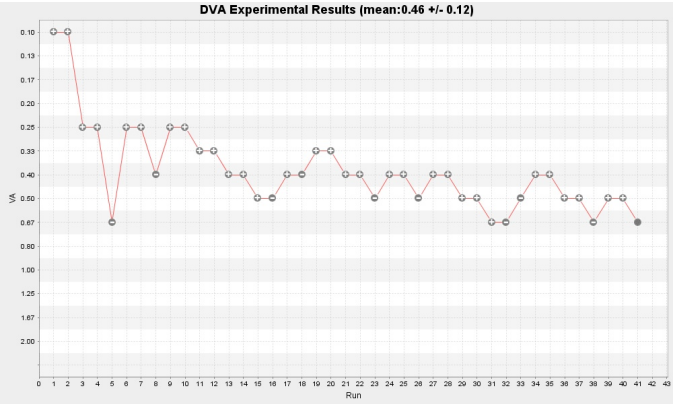


Figure 10: Right eye, 0 km/h

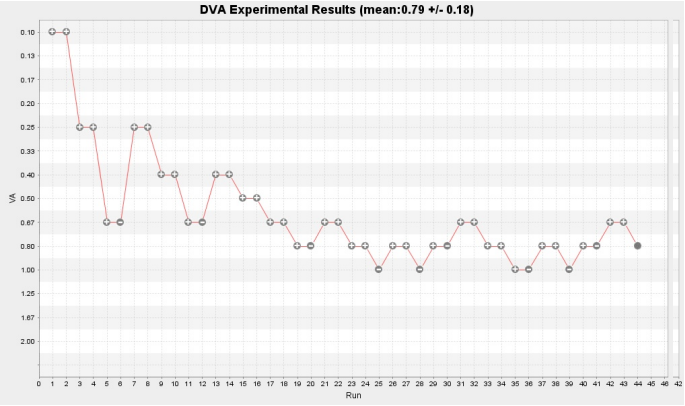


Figure 12: Left eye, 9 km/h

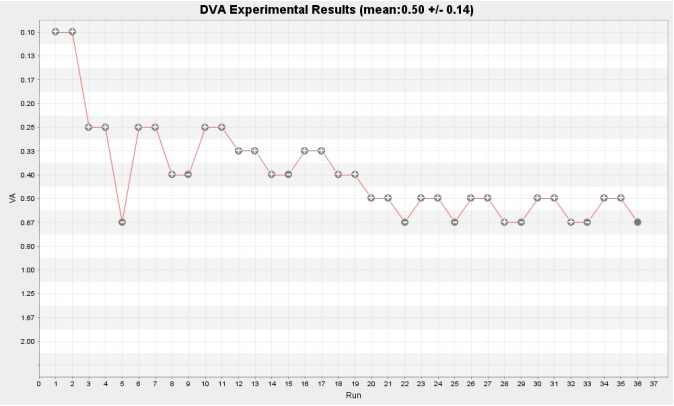


Figure 13: Right eye, 9 km/h

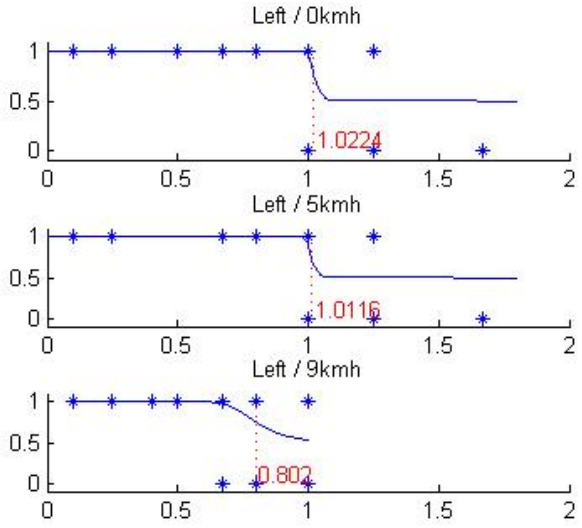


Figure 14: Psychometric function, left eye

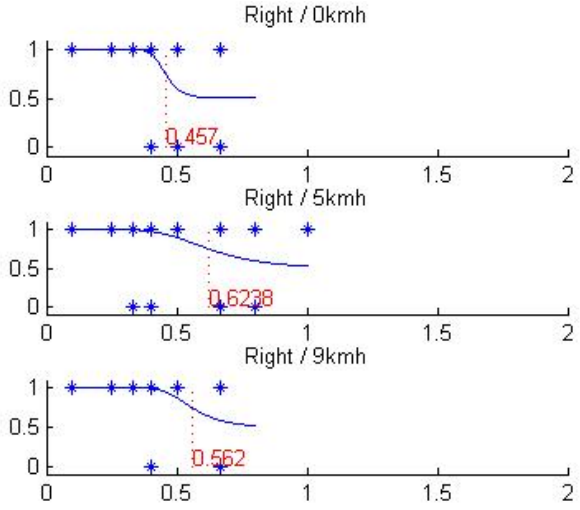


Figure 15: Psychometric function, right eye

2)Individual 2 – W, 21, glasses

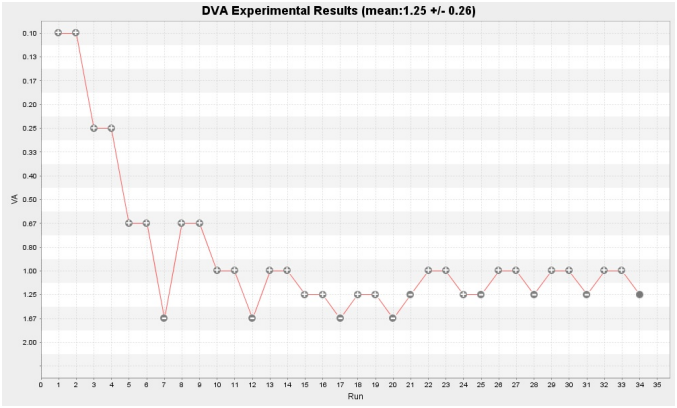


Figure 16: Left eye, 0 km/h

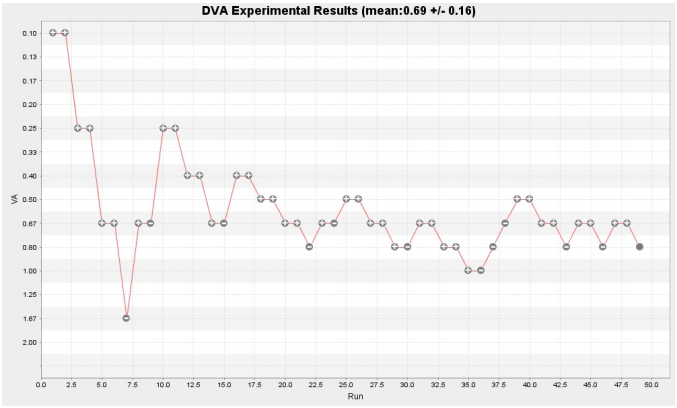


Figure 17: Right eye, 0 km/h

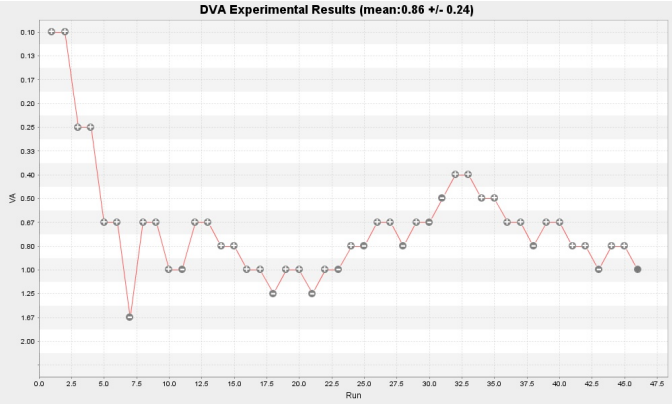


Figure 18: Left eye, 5 km/h

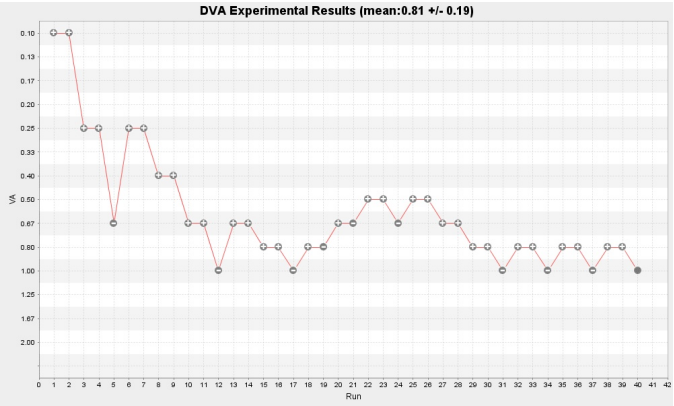


Figure 19: Right eye, 5 km/h

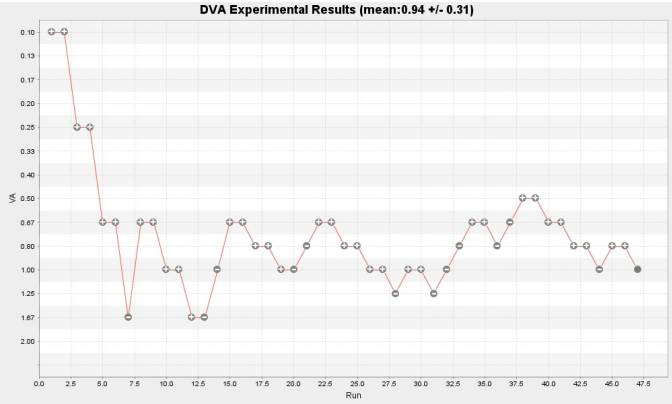


Figure 20: Left eye, 9 km/h

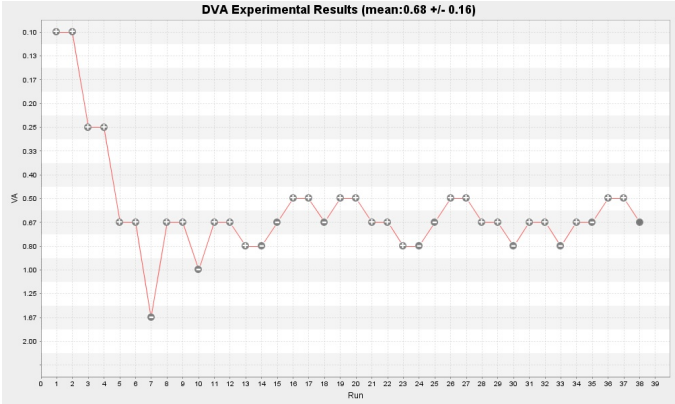


Figure 21: Right eye, 9 km/h

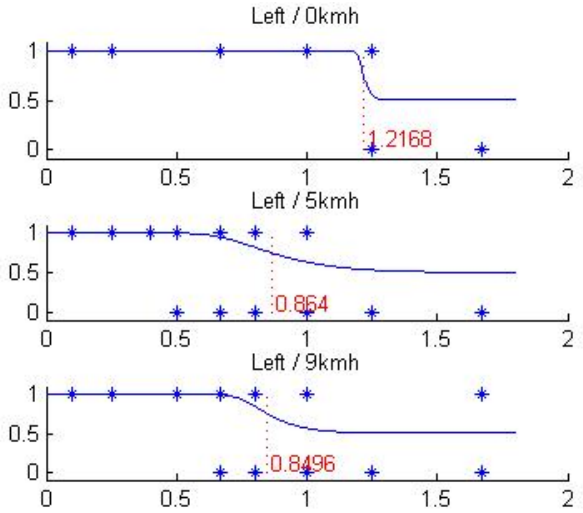


Figure 22: Psychometric function, left eye

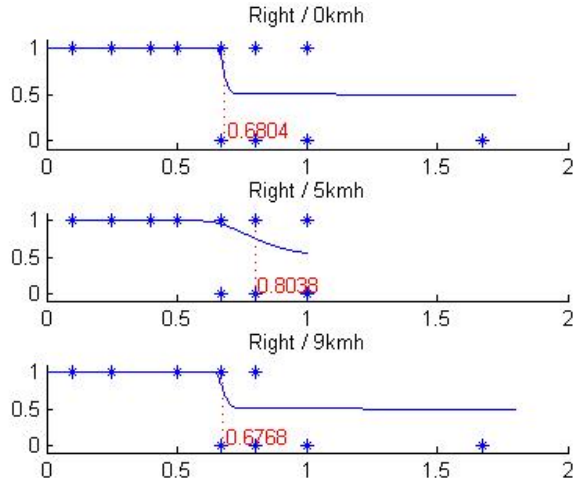


Figure 23: Psychometric function, right eye

3)Individual 3 – M, 23, lenses

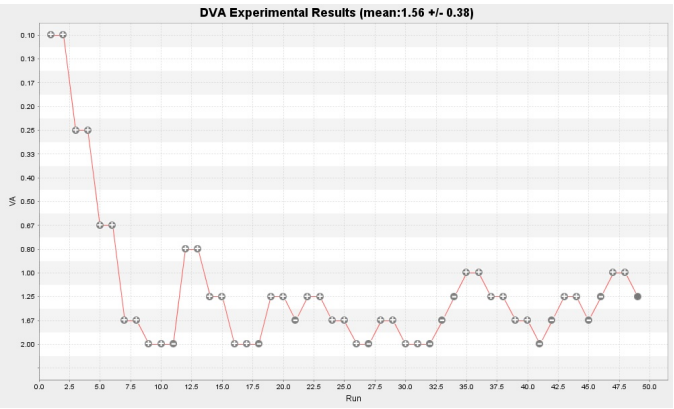


Figure 24: Left eye, 0 km/h

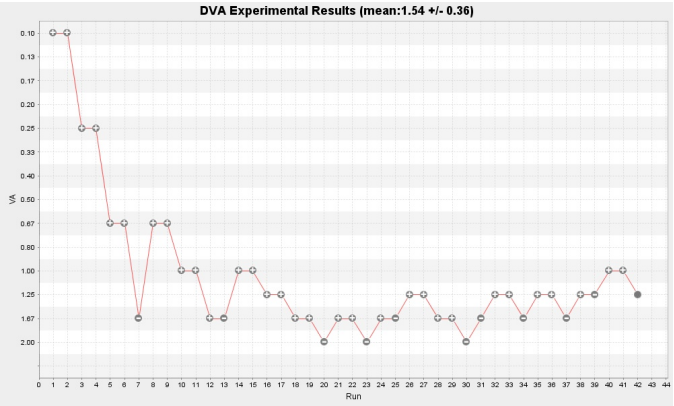


Figure 25: Right eye, 0 km/h

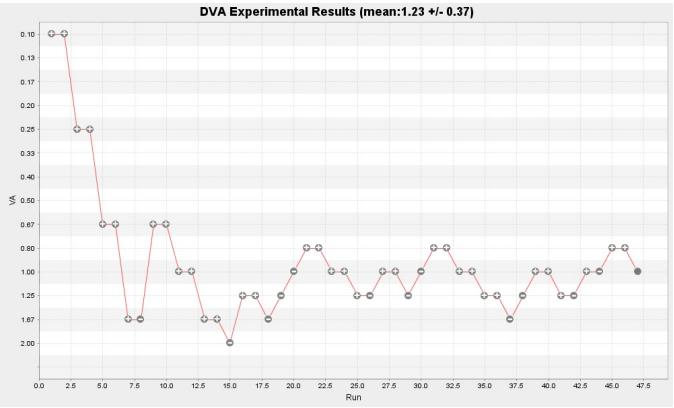


Figure 26: Left eye, 5 km/h

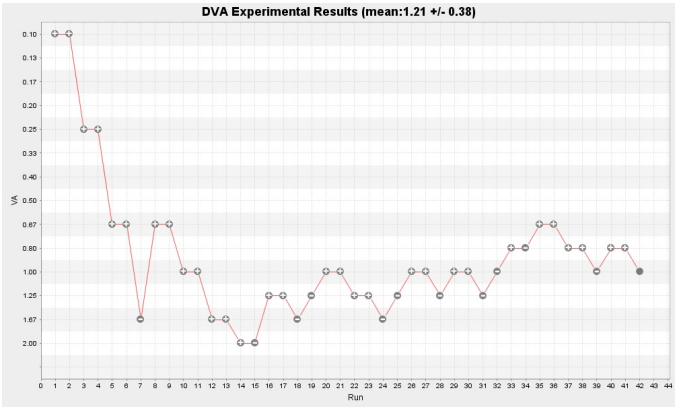


Figure 27: Right eye, 5 km/h

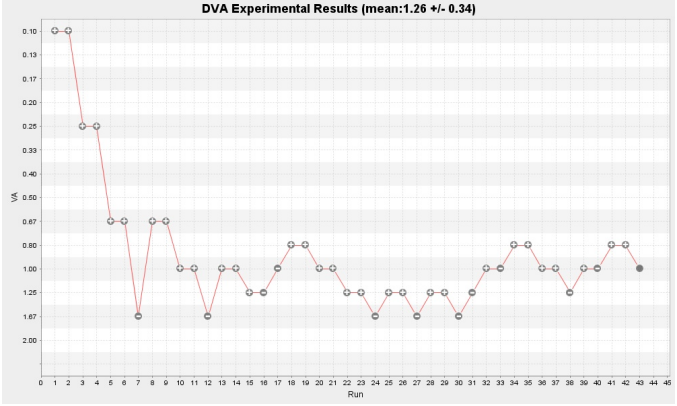


Illustration 28: Left eye, 9 km/h

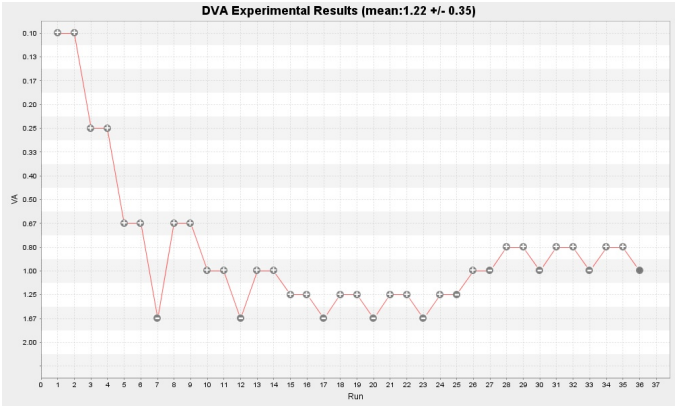


Figure 29: Right eye, 9 km/h

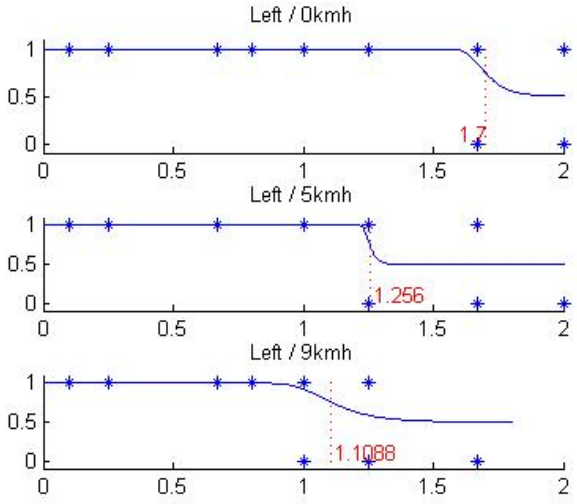


Figure 30: Psychometric function, left eye

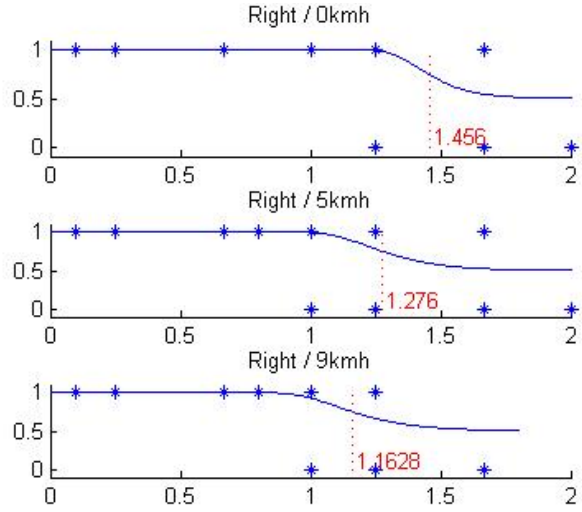


Figure 31: Psychometric function, right eye

4)Individual 4 – M, 23, glasses

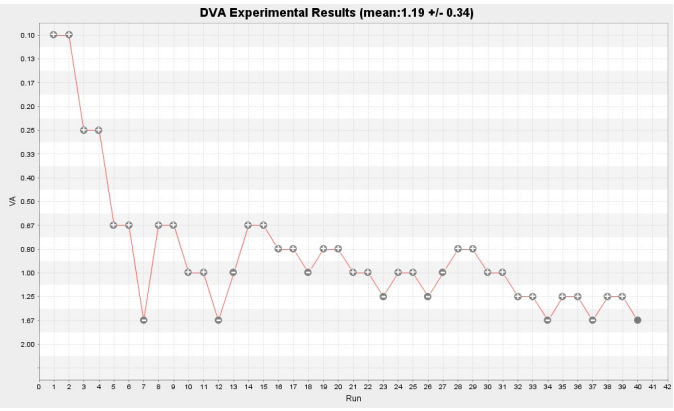


Figure 32: Left eye, 0 km/h

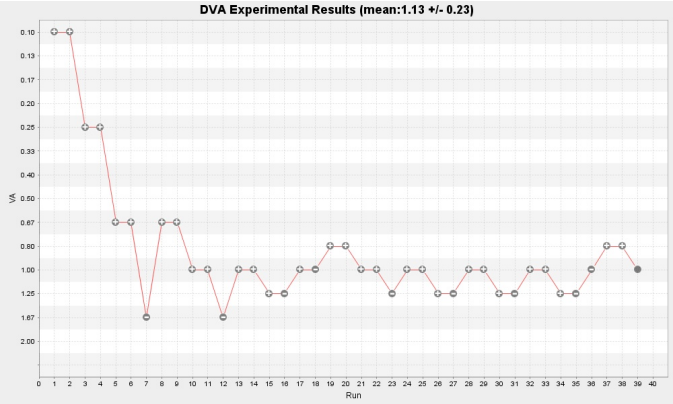


Figure 33: Right eye, 0 km/h

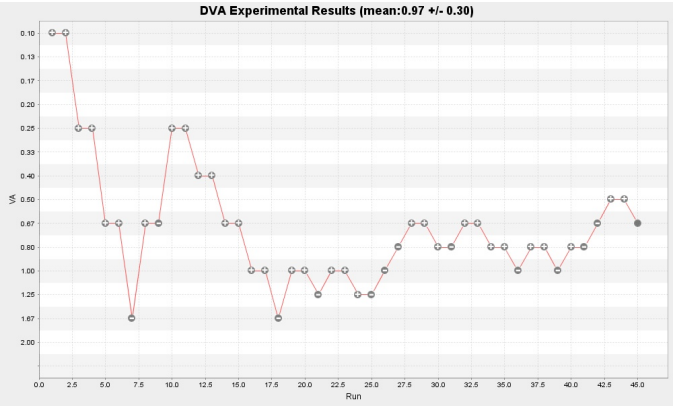


Figure 34: Left eye, 5 km/h

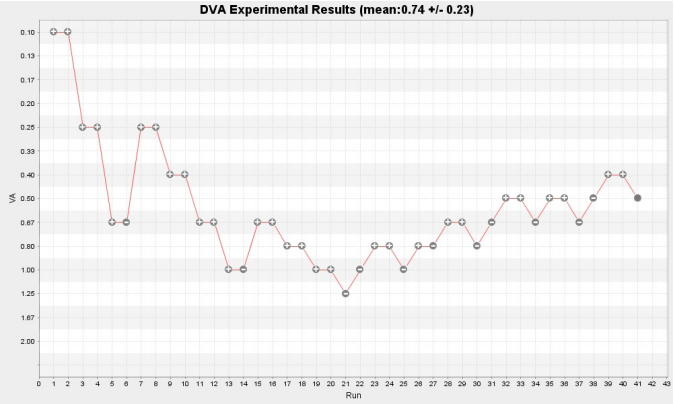


Illustration 35: Left eye, 9 km/h

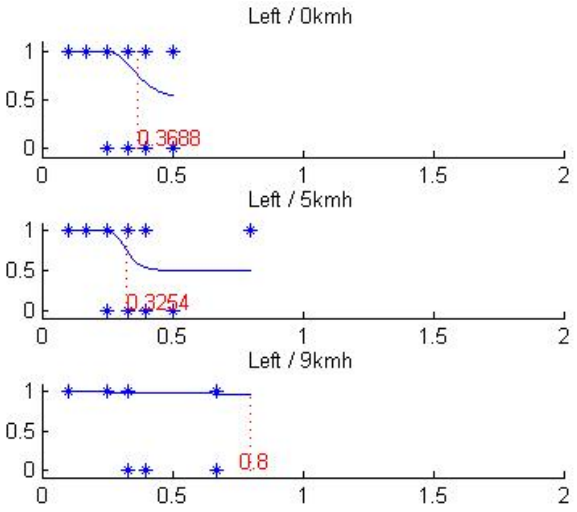


Figure 37: Psychometric function, left eye

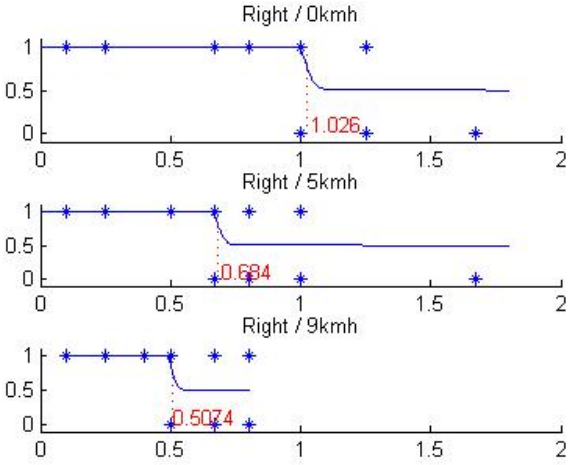


Figure 38: Psychometric function, right eye

5)Individual 5 – M, 24, glasses

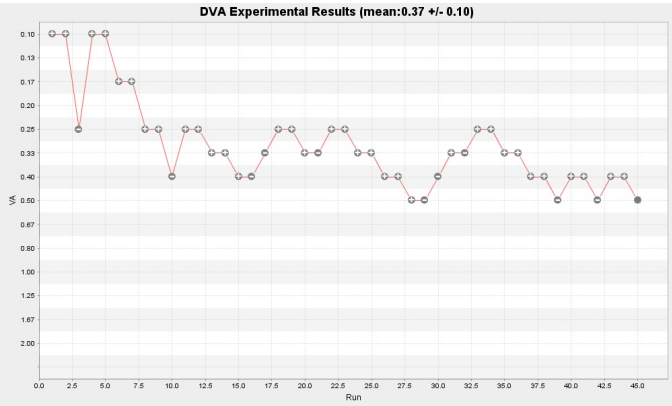


Figure 39: Left eye, 0 km/h

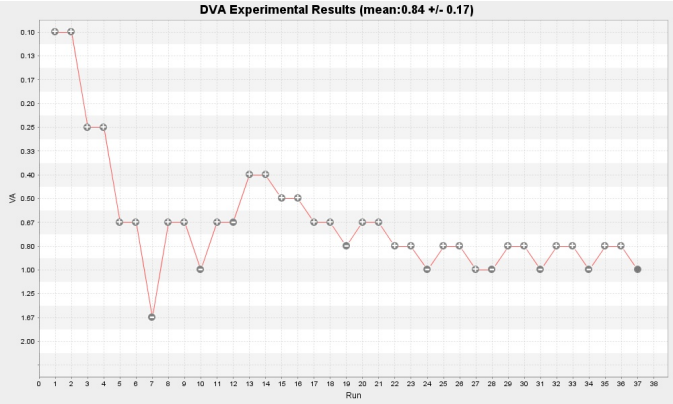


Figure 40: Right eye, 0 km/h

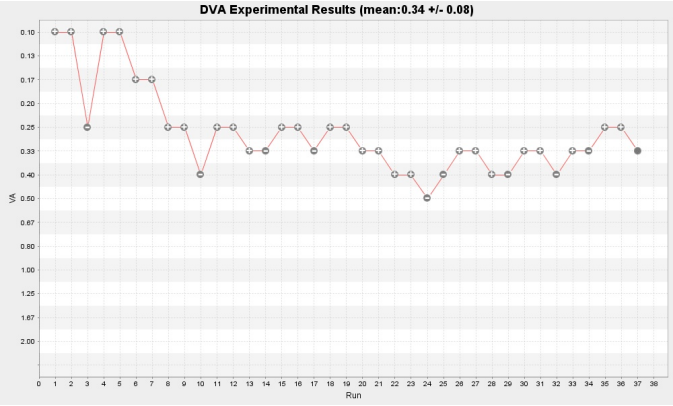


Figure 41: Left eye, 5 km/h

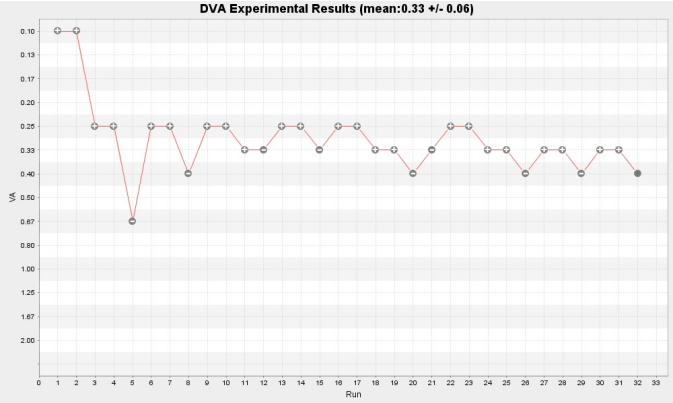


Figure 42: Left eye, 9 km/h

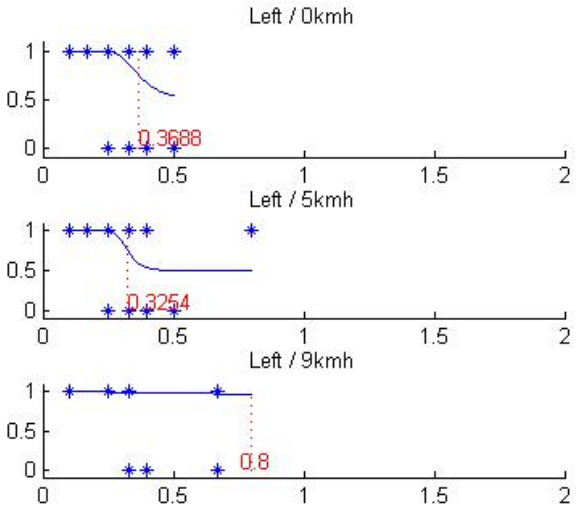


Figure 44: Psychometric function, left eye

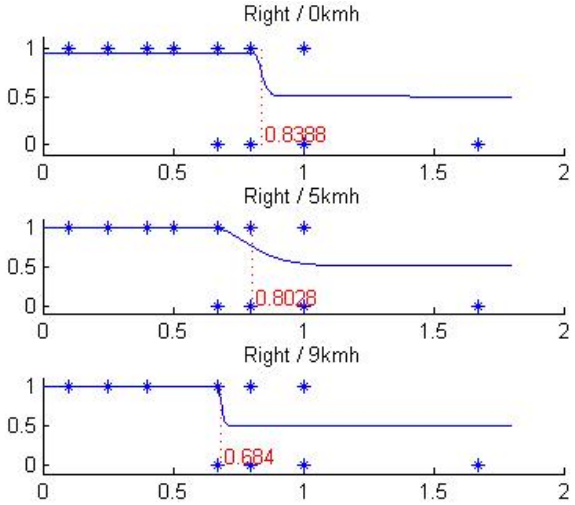


Figure 45: Psychometric function, right eye