VizFix Software Requirements Specifications

Tim Halverson and Anthony Hornof
Computer and Information Science
University of Oregon
5/21/02 - DRAFT

This is a proposed software requirements specification for VizFix--a tool for viewing and analyzing eye movement data.

Problem Statement

Eye tracking technology is becoming increasingly popular and useful as a behavioral measure of how people interact with computers, and as an independent measure to assist in the evaluation of interfaces. Eye trackers have become increasingly inexpensive and easy-to-use. Many HCI research labs that only peripherally deal with visual issues in their research are now buying eye trackers.

Eye tracking technology is becoming increasingly easy to set up and to use. However, the capture, representation, and analysis of eye movement data remains a difficult and challenging task. There are many steps involved in the processing of the eye movement data between the moment that the image of the moving eye enters the camera and the moment at which a researcher can make a well-supported claim about the patterns of eye movements that were observed or the cognitive strategies that motivated the eye movements.

In short, the five steps involved in the automated capture, representation, and analysis of eye movement behavior can be summarized as thus: (1) collection of gaze samples, (2) fixation identification, (3) fixation visualization, (4) post hoc error correction, (5) data categorization, coding, and classification, (6) developing and evaluating theories of eye movements. These steps are enumerated in greater detail in Usage Scenario 1.

Step #1, the image processing of video frames to produce gaze samples, is continually improved by eye tracking manufacturers. The algorithms, techniques, and source code are usually proprietary. Step #2, fixation identification, is a reasonably well-solved problem. Algorithms are well-established and are readily available. Though the debate over what exactly constitutes a fixation continues, and different algorithms with different parameter settings will identify slightly different fixations.

General purpose solutions to #3, #4, and #5, however, are not readily available. Fixation visualization (#3) is key because it gives the analyst the ability to look at eye movement scan paths and apply his or her powerful human pattern-matching capabilities, looking for trends in the data, and even identifying the extent to which the data may be affected by constant or variable error. Good visualization tools gives the analyst the first-pass opportunity to make sense of the eye movement data, the way that a good statistical analysis and graphic package allows an analyst to make multiple passes of analysis over reaction time and other observed data. Because eye movement data is always spatial, because eye movement data is inherently noisy, and because there is almost always a visual stimuli involved in the task, looking at the scan patterns superimposed on that stimuli is a very important step in the analysis process.

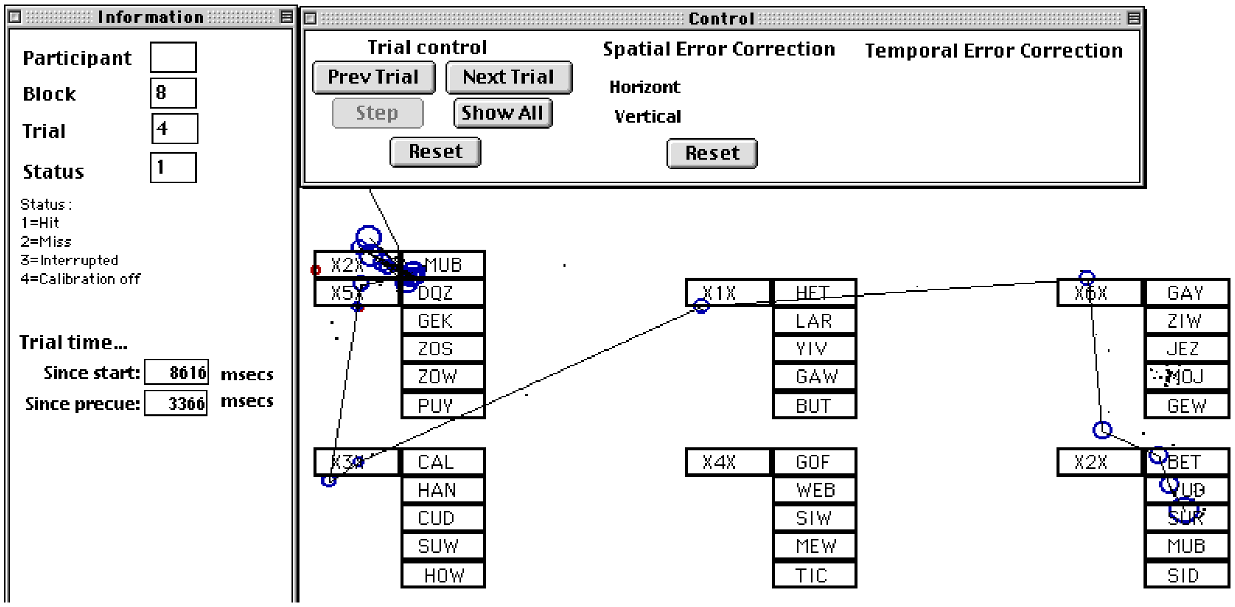
Usage Scenarios

Scenario 1. Categorizing Visual Search Patterns in Hierarchical Layouts

This is a very general description of one problem the tool will be used to solve.

Computational cognitive models were built to investigate how people search labeled visual hierarchies--in which every group of items in a visual layout has a heading that indicates if the target item is in that group (Hornof, in preparation; Hornof, 2001). A two-tiered search strategy was proposed because, given the constraints associated with human visual search processes as captured by the EPIC cognitive architecture (Kieras & Meyer, 1997), the only way the participants could have found the targets as quickly as they did was to conduct a two-tiered search--first searching group labels, and then searching within the target group. A most cursory review eye movement data collected for this task validates this theory--in nearly all cases, participants first moved their eyes among the group labels, and then searched within the target group (unpublished data from work in progress). Figure 1 shows a visualization of eye fixations from the experiment. Numerous similar traces demonstrate that participants conducted a two-tiered search: They first used the group labels to find the target group, and then searched for the target item exclusively within that group.

To arrive at the view of the data that allows us to conclude that people conducted a two-tiered search, substantial automatic capture, representation, and analysis of user behavior was required. First, the eye tracker and computer sampled the gaze location every 17 msec. Second, the gaze data was automatically converted into fixation position and duration data. Third, the fixations and durations were superimposed on the viewed scene in order to reveal the general trends and systematic errors in the data.


Information

Participant:
Block: 8
Trial: 4
Status: 1

Status:
1=Hit
2=Miss
3=Interrupted
4=Calibration off

Trial time...
Since start: 8616 msecs
Since precue: 3366 msecs

Control

Trial control
Spatial Error Correction
Horizont
Vertical
Temporal Error Correction

Figure 1. Screenshot from VizFix, the eye fixation visualization and error correction tool currently being developed by Hornof and Halverson, showing a visualization of eye fixations from a visual search experiment. The lines connect consecutive fixations, with the diameter of each circle indicating the duration of the fixation. In the experiment, the participant studied the precue (X2X MUB, in the top left corner), clicked the mouse on the precue to make the layout appear (though both the precue and layout are shown here together), and found and clicked on the target item as quickly as possible. In this trial, the target is the item MUB in the group labeled X2X, at the bottom left of the layout. The trace of the eye movement data shown here demonstrates that, when group labels (X1X, X2X, etc.) were shown, participants used them to first find the target group, and then searched within that group.

These three automatic capture and representation steps were sufficient to reveal the two-tiered search in the data for labeled hierarchies. However, two additional steps are required to measure and automatically analyze and categorize the eye movement patterns for a single trial as following a systematic two-tiered search. Fourth, systematic errors in the data must be automatically identified and corrected. Note in Figure 1 that the fixations at the bottom right appear to be roughly one item above the items that were probably fixated (such as the *X2X* and *MUB*). This is probably due to systematic error in the eye tracking data. The data should be thrown out, systematically adjusted, or considered in terms of relative eye movements and not absolute position on the layout.

Fifth, the fixation data must be systematically categorized so that the theory can be tested. To test for a two-tiered search strategy, the automated categorization should be relatively straightforward: Did the eyes first look at group labels and then within the target group? But there are roughly 2,000 trials for which this question needs to be answered. Automated visual search strategy identification tools are needed. Salvucci and Anderson's (2001) automated protocol analysis may be overwhelmed by the many different eye movement patterns that would correspond to a two-tiered search, and may need to be extended.

The theory and the data interact. Data must be collected and analyzed to test the theory, and the theory must lend itself to evaluation with the data. Byrne (2001) and Anderson et al. (1998) proposed feature-based menu searches, in which the eyes moved to menu items that had a similar shape as the target. The models were evaluated with respect to how well they predicted reaction time data and general patterns of eye movements (Byrne, 2001; Byrne, Anderson, Douglass & Matessa, 1999), but not with respect to the central assumption of the model that the eyes tended to move to items that shared the same shape as the target. Such an analysis would be quite involved, but would ultimately be the best way to test the theory. Each item in the layout would need to be categorized with respect to its similarity to the target, and every fixation would need to be categorized with respect to whether it was to an item that shared features with the target. Existing automated eye movement protocol analysis will be useful only to the extent that a limited number of well-defined eye movement patterns can be identified in advance. It is likely that new automated eye movement protocol analyses will need to be developed.

Other specific questions we would like to answer about the data collected in this experiment, which may require additional software beyond VizFix:

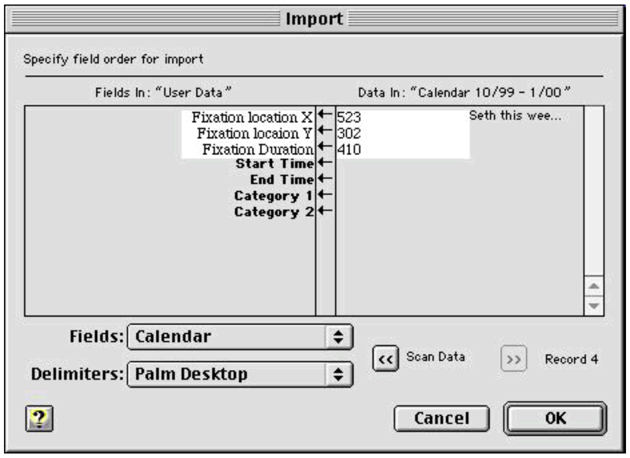
- Determine the feature-similarity between every object and the target, and determine to what extent fixations landed on items similar to the target.
- When did people move their eyes from the precue in relation to the moment that the mouse button was clicked?
- When did the mouse start to move in relationship to when the gaze was at the target? Do a separate analysis for trials on labeled layouts, to see when the mouse started to move in relationship to when the target group label was fixated. Can we identify a number of successful trials in which the mouse started to move before the target was fixated? How often did the point-completion deadline catch these trials? How many slipped through? How long after the eye fixated the target did the mouse started to move in relationship to when the gaze fixated the target group label?
- How often if ever did the gaze land at the target, but the target was missed?

Scenario 2: On-the-spot visualization of another researcher's data.

This is a sort of "fantasy" scenario.

Anthony is attending the ETRA 2002 symposium and watches Joe Goldberg's presentation in which he shows some screenshots with eye movement data superimposed. Anthony recognizes the potential value of the data as a useful data set for helping to drive the development of algorithms and a tool for the automated analysis of and strategy-extraction from eye movement data. One important step in using the data for this purpose would be to actually look at the eye movements as they unfold, superimposed on the stimuli. The human analyst could look for eye movement patterns that suggest strategies, and could assess the extent to which the data is "clean" or contains a lot of error. Anthony asks Joe if he could have a copy of his data. Joe says sure and that, in fact, he has it on his laptop. Somehow they get a copy of the data and the screenshots used in the task onto Anthony's iBook, and they proceed to analyze it.

First, Joe explains to Anthony the format of the data files. This could be anything. The data could be tab-delimited, carriage-return-delimited, space-delimited. The data could be broken up by individual samples at a variety of sampling rates, or by fixation with a variety of recorded details, or perhaps at some other grain size. The data could be broken up into blocks, or not, with a variety of indications of when the blocks (or *time spans*?) start and end. Anthony somehow quickly enters these parameters into VizFix. Perhaps he types some parsing strings using regular expressions, or perl. Perhaps he just selects "Import..." from the file menu and sees a dialog box like that shown in Figure X, but with fixation data parameters such as "fixation duration" showing instead of "Title", and with the various numbers listed out on the right. Joe could just point at the numbers and say which is which. Anthony could line the numbers up with the various parameters, and then commence the import. At some point in the import process, Joe would also tell Anthony the units of measurement of the data (*e.g.*, screen pixels, mm, etc.), and Anthony would somehow enter these parameters as well.


Import

Specify field order for import

Fields In: "User Data" Data In: "Calendar 10/99 - 1/00"

Fixation location X ← 523 Seth this wee...
Fixation locaion Y ← 302
Fixation Duration ← 410
Start Time ←
End Time ←
Category 1 ←
Category 2 ←

Fields: Calendar
Delimiters: Palm Desktop

Figure X. Perhaps the fixation data import function will resemble the calendar import function on Palm Desktop 2.6.3.

Once the data format is identified, and the fixation data imported, the screenshots used in the task still need to be imported so that the fixations can be superimposed on the screenshots. Perhaps ideally the software package would import and display a screenshot image in any graphics file format, but the conversion could also be done with other applications such as Photoshop or conversion utilities. Besides importing the graphics file, a few parameters associated with the file are also imported, such as the size and scale of the drawing, and how the image relates to the eye movement data. For example, if the data is in *(x, y)* coordinates, VizFix needs to know which point on the drawing is (0, 0). Potentially, Joe's screenshot is even bigger than the iBook screen, and needs to be reduced to fit on the screen.

Joe and Anthony start looking at the scan paths, and they see some constant error. All of the fixations near the top of the image seem to be recorded a little to the left, and at the bottom of the screen a little to the right. Anthony draws two error vectors, one at the top of the screen pointing to the left, and one at the bottom of the screen pointing to the right (or he draws four, one in each corner). He then turns on the error-correction transformation function, and all of the fixations are shifted proportionally to their distance from the vector. Alternately, if he can't draw the vectors directly, he can at least move the mouse around the screen so that he can see all of the pixel location coordinates, decide exactly where he wants to draw the vectors, enter the vectors into an input file, and then see the vectors on the screen.

They save all of their work, including the importing functions necessary for this data file, the two-vector error signature that they determined while looking at the data, ... What is more, they save it so that it is automatically associated with the original input data file the next time that file is opened. (Perhaps this is done by creating a project folder, as in CodeWarrior (CW) or other IDEs, though perhaps the overhead in learning how to work a system based on project folders would prohibitively raise the learning curve involved in using VizFix. It takes a little while to get used to the CW project folders. The project folders also sort of create a little bit of extra "junk" in your file system.)

Additional fantasies:

- Re-create the temporal aspects of the task.
- Mouse movement data.
- Then they view the data, somehow enter in some strategies, and then automatically parse the data based on these strategies. They highlight subsections of movements.
- Looking at eye movement data, we see two fixations where we expect to see one, or one where we expect to see zero, or zero where we expect to see one. We wonder if slightly different settings of the fixation detection algorithm would produce different claims about fixation locations. While still looking at the data, we change the settings the the algorithm and replot the data. We then go back and forth between the two different plots. We then plot all of the individual samples, and overlay the fixations on top, going back and forth between just the samples and the samples with the fixations.

Scenario 3: Random person wants eye tracking done on some image.

Basically use the Gazedemo software like we have with Richard Taylor and Co., but so that it does everything we wanted it to do while they were visiting.

Could we have VizFix interact directly with the same s/w as the user?

Requirements

Functional Requirements

Note: The highest level entries are user requirements. Lower level entries are system requirements.

1. One or more eye-tracking experiment data files may be open at a time.
   - 1.1. Each data file shall be displayed in a separate window.
   - 1.2. The title of each window shall be the participant number of the participant from which the data was collected.
   - 1.3. A windows menu shall list all open window.
   - 1.4. The open-file dialog box shall allow for the selection of multiple files to open.

2. Visualization of raw gaze samples shall be available.

3. Visualization of fixations shall be provided.
   - 3.1. The temporal order of fixations shall be indicated by color or labels. The indication method shall be user configurable.
   - 3.2. Various methods of fixations analysis will be supplied. At a minimum, velocity-threshold and displacement-threshold algorithms will be available.
   - 3.3. The parameters for the fixation analysis algorithms shall be dynamically adjustable.

4. Visualization of fixations corrected for systematic error shall be available.
   - 4.1. Various methods to correct fixation locations for systematic error shall be provided.
   - 4.2. A vector representation of the correction may be displayed.

5. Visualization of search strategies shall be furnished.
   - 5.1. Various algorithms shall be supplied for the determination of search strategies.

6. The cursor position shall be able to be displayed.

7. Gaze point samples and aggregate data for the gaze point samples shall be simultaneously displayed. The user shall be able to dynamically determine which data is shown.
   - 7.1. Pull-down menus, or the like, shall be provided for each type of data (raw, fixation, corrected fixations, and strategy) to dynamically change if that data is displayed and which algorithm is used in the analysis.
   - 7.2. A preferences pane will specify the default information visualization to be displayed, along with any variable parameters.

8. Temporal components of experiments shall be displayed synchronized with the eye gaze data.
   - 8.1. The system shall be able to display events including, but not limited to, mouse clicks, dialog boxes, and changes in experimental stimuli.

9. The user shall be able to navigate through the temporal hierarchy of the experimental data.
   - 9.1. The system shall supply a means to skip forward or backward one temporal unit at each level of the experiments temporal hierarchy.
   - 9.2. A means of skipping to the beginning of each level of the experiments temporal hierarchy shall be

provided.

9.3. At the lowest level of the experiments temporal hierarchy, the user shall be able to step sequentially through individual fixations.

10. The user shall be able to view textual summary information of the data files.
    - 10.1. The system shall display the total number of fixations at any specified level of the experiments temporal hierarchy.
    - 10.2. The system shall display the average fixation time at any specified level of the experiments temporal hierarchy.
    - 10.3. The system shall display a list of fixations at any specified level of the experiment's temporal hierarchy.
        - 10.3.1. A single fixation may be selected from the list to display summary data for that fixation including:
            - 10.3.1.1. Fixation duration
            - 10.3.1.2. Fixation start time.
            - 10.3.1.3. Fixation end time.
            - 10.3.1.4. Area of interest that was fixated, if areas of interest are defined.
    - 10.4. The system shall display the total and elapsed time for each level of the experiments temporal hierarchy, with the current step in the lowest level of the hierarchy as the end point for the elapsed time.
    - 10.5. The system shall provide a preferences pane to determine which of the above summary information should be displayed by default.

11. The user shall be able to manually correct for spatial error.
    - 11.1. This correction will be only for constant horizontal and vertical error.
    - 11.2. The degree of correction (in pixels, inches, or other measures of distance) shall be displayed for horizontal and vertical.
    - 11.3. The user shall be able to reset the manual correction to zero.

12. The user shall be able to set a global temporal correction between gaze samples and other data to compensate for system delays.

13. The user shall be able to manually correct for temporal error at the lowest level in the temporal hierarchy on a per segment basis.

14. The user shall be able to export extracted data given all possible outputs/views described in 3 through 5.
    - 14.1. The user shall be able to select the type of data exported for each export.
    - 14.2. Various types of delimited text files shall be supported.
    - 14.3. The system shall provide a preferences pane to set the default data to export.

15. The user shall be able to display either dynamic or static images or layouts as backdrops for the gaze information in each window.
    - 15.1. If the data file specifies the backdrop for any level in the experiments temporal hierarchy, the system shall be able to display it.
    - 15.2. The user may manually select images, on a per window basis, for use as a backdrop to the gaze information.
    - 15.3. The system shall support the following file formats for backdrop images: PICT, GIF, JPEG, BMP, TIFF, and PNG.

Non-functional Requirements

1. Product requirements
    - 1.1. The system shall accept a variety of data formats for experiment data input.
    - 1.2. The system shall be able to adjust to a variety of screen sizes and resolutions.
    - 1.3. The system shall not change any data in the input file.
    - 1.4. Input and output files shall use ASCII format.

    1.5. The Macintosh version of the software shall be optimized for PPC (non-CPU specific).
    1.6. The Macintosh version of the software shall be Carbon-compliant.
    1.7. "ONR Inside" will appear on the splash screen when starting the program, and the "about" box.

2. Organizational requirements
    2.1. The system shall be implemented in C++.
    2.2. The system shall be implemented using CodeWarrior.
    2.3. The system shall be implemented using the object-oriented paradigm.
    2.4. The objects that use OS specific methods shall be kept to a minimum to allow maximum portability.

3. External requirements
    3.1. The system shall never display confidential information about experiment participants.
    3.2. The application's splash screen and about box shall list the source of grants that funded the development.

Standard File Format for Eye Tracking Experimental Data

string - an arbitary string of characters in ASCII
int - an integer in ASCII
{a, b, c} - a list of valid values

```
<p>     // Beginning of participant info
        <ident> string</ident>// Identifier, e.g. participant number
        <date>string </date>            // Date of the experiment
        <time>string </time>            // Starting time of the experiment
        <units> int</units>             // The depth of the experiments temporal heirarchy.
        <cref>{center, topleft</cref>   // The reference point for coordinates
        <cond>string </cond>// Any experimental condition information
</p>

<tu>    // Beginning of a new temporal unit
        <tu> // The depth of the temporal unit markers must match the value for <units>
        .
        .
        </tu>
</tu>   // End of coresponding temporal unit

<g>     // Beginning of a gaze and cursor position sample
        <f>{0,1}</f>                    // Indication if gaze sample was found. 0=no, 1=yes
        <gx>int </gx>           // X-coordinate of the gaze sample
        <gy>int</gy>                    // Y-coordinate of the gaze sample
        <gp>int</gp>            // Pupil size
        <my> int</my>                   // x-coordinate of the mouse sample
        <my> int</my>                   // Y-coordinate of the mouse sample
</g>
```

```
<aoi>   // Area of interest.  Exists for the duration of the temporal unit in which it is defined.
        <shape>{rect, circle}</shape>           // Shape of the AOI
        <x>int</x>                              // X-coordinate of the center
        <y>int</y>                              // Y-coordinate of the center
        <rad>int</rad>                          // Radius of the circle
        <l>int</l>                              // Length of the rect
        <w>int</w>                              // Width of the rect
        <text>string</text>                     // Text in the AOI
<aoi>

<ev>    // Event
        <type>{click, dialog, scroll, image, sound}</type>
                                                // Click = mouse click; dialog = dialog box; scroll=
                                                // page scroll; image = screen image changed;
                                                // sound = sound played
        <x>int</x>                              // X-coordinate of the center of the event
                                                // Valid for click or dialog
        <y>int</y>                              // Y-coordinate of the center of the event
                                                // Valid for click or dialog
        <dir>{up, down, left, right}</dir>      // Direction of scroll
        <desc>string</desc>               // Description
                                                // The text of the dialog, or the identifier of the
                                                // image or sound file.
</ev>
```