The Eyelink Toolbox:

Eye Tracking with MATLAB and the Psychophysics Toolbox.

Frans W. Cornelissen[1], Enno M. Peters[1], John Palmer[2]

[1]Laboratory of Experimental Ophthalmology, School for Behavioral and Cognitive Neurosciences (BCN)

University of Groningen, PO Box 30001, 9700 RB Groningen, The Netherlands.

[2]Psychology, University of Washington, Box 351525, Seattle WA 98195-1525, USA

Running head: The Eyelink Toolbox

For the Eye Tracking issue

Corresponding author:

Frans W. Cornelissen

Laboratory of Experimental Ophthalmology

School for Behavioral and Cognitive Neurosciences (BCN)

University of Groningen

PO Box 30001, 9700 RB Groningen

The Netherlands

Tel: +31 50 3614173

Fax: +31 50 3611709

E-mail: f.w.cornelissen@med.rug.nl

Abstract

The Eyelink Toolbox software supports the measurement of eye movements. The toolbox provides an interface between a high-level interpreted language (MATLAB), a visual display programming toolbox (Psychophysics Toolbox) and a video-based eyetracker (Eyelink). The Eyelink Toolbox enables experimenters to measure eye movements while simultaneously executing the stimulus presentation routines provided by the Psychophysics Toolbox. Example programs are included with the toolbox distribution. Information on the Eyelink Toolbox can be found via http://psychtoolbox.org/.

Keywords

Eye-movements, psychophysics, MATLAB, Psychophysics Toolbox, software

Introduction

Measuring eye-movements during psychophysical tasks and experiments is important for studying eye-movement control, gaining information about a level of behavior generally inaccessible to conscious introspection, examining information processing strategies, as well as controlling task performance during experiments that demand fixation or otherwise require precise knowledge of a subject's gaze direction (e.g. Cornelissen & Dobbelsteen 1999, Brenner & Cornelissen 2000). Eye-movement recording is becoming a standard part of psychophysical experimentation.

Whereas eye-tracking techniques exist that rely on measuring electrical potentials generated by the moving eye (electro-oculography) or a metal coil in a magnetic field, such methods are relatively cumbersome and uncomfortable for the subject (e.g. because electrodes have to be attached to the head or a coil has to be placed on the cornea). A new generation of eye-trackers is now available based on the non-invasive recording of images of a subject's eye using infra-red sensitive video-technology and relying on software processing to determine the position of the subject's eyes relative to the head. As these trackers are video-based, there is no need for direct contact with the subject's eyes, making these trackers much more suitable for routine eye-movement recording during longer sessions. By combining the information on eye-position with a measure of head-position, estimates of gaze position on a display can be obtained allowing the creation of gaze-dependent displays.

Eyelink Gazetracker

The Eyelink Gazetracker (SR Research Ltd., Mississauga, Ontario, Canada) is one of these video-based eye trackers and is used in research fields such as psychology, ophthalmology, neurology, and ergonomics. The Eyelink uses two high-speed cameras (CCD sensors) to track both eyes simultaneously. A third camera (also a CCD sensor) tracks four infrared markers mounted on the visual stimulus display, so that head motion can be measured and gaze position can be computed. The cameras produce images at a sampling rate of

250 Hz (4 ms temporal resolution). The Eyelink uses a PC with dedicated hardware for doing the image processing necessary to determine gaze-position. Pupil position is tracked by an algorithm similar to a centroid calculation, with a noise-limited resolution of 0.01° or less, and velocity noise of less than 3 deg/sec (manufacturer's specifications). An optional heuristic filter (Stampe 1993) removes single-sample noise artifacts and does not affect measured saccade velocity and acceleration. Pupil position is mapped to a head-referenced coordinate system by a biquadratic mapping function (Sheena & Borah 1981, Stampe 1993) for two-dimensional eye tracking, or by a quadratic function for one-dimensional eye tracking. Head position data is then combined with the head-referenced data to compute the true gaze position on the stimulus display. The optical head tracking system has extremely low angular noise, and therefore does not appreciably increase the final noise level of the system.

The Eyelink communicates via a high-speed ethernet connection with a separate computer (Macintosh or PC) that performs the stimulus display. The Eyelink system parses the eye-movement data on-line and in real-time so that information on eye position is available almost instantaneously. Average delays from eye movement to position data availability are 6 ms with heuristic filtering disabled, and 10 ms with filtering enabled (manufacturer's specifications). This allows the creation of (near) real-time gaze contingent applications (e.g. Cornelissen & Dobbelsteen 1999, Tant, Cornelissen, Brouwer & Kooijman 2002). The parser also provides information on fixation and saccade events and associated statistics immediately after the events have been completed.

One limitation of the EyeLink is that there is considerable trial-to-trial variability in the estimate of absolute gaze position. For example, typical measurements of the standard deviations of gaze position at fixation are 0.5° horizontal and 1.5° vertical. This variablity may be due to slippage of the headmounted system over time. This problem can be minimized by using fixation position at the beginning of each trial to "correct" the gaze estimate (“drift correction”). This method provides good relative position estimates within a trial at the cost of information about absolute position across trials.

Stimulus presentation

Most eye-movement experiments require displaying visual targets for the subject to track or look at. Computer displays have become virtually standard equipment for doing visual psychophysics since these allow precise software specification of the stimulus. While low-level programming languages such as C enable the required accuracy in control of timing, luminance and color output of displays, they do not provide a friendly programming environment. The Psychophysics Toolbox (Brainard 1997, Pelli, 1997) is a software package that adds the ability for precise stimulus specification to MATLAB, a high-level interpreted language with extensive support for numerical calculations (The MathWorks, 1993), allowing for rapid and flexible programming of psychophysical experiments.

We have developed the Eyelink Toolbox, a high-level interface between MATLAB and the Eyelink Gazetracker. The toolbox enables one to measure eye movements while simultaneously executing stimulus presentation routines provided by the Psychophysics Toolbox as well as other MATLAB scripts. The powerful combination of the Psychophysics and Eyelink Toolboxes allows for a relatively fast and easy implementation of experiments involving eye tracking (such as e.g. gaze-dependent displays).

Implementation of the toolbox

The toolbox consists of a combination of a MATLAB extension (MEX) file and code written in native MATLAB language. The extension, which is partly based on proprietary low-level C subroutines provided by the manufacturer of the eyetracker, can be called directly from MATLAB. For its timing and graphics manipulation, the Eyelink Toolbox relies on routines of the Psychophysics Toolbox (Brainard 1997) and VideoToolbox (Pelli, 1997).

The Eyelink Toolbox provides access to all Eyelink routines. A listing of the main commands and a short description of their function is provided in table 1. In addition, the Eyelink Toolbox provides both integrated routines for performing calibration and drift correction, as well as procedures written in native MATLAB

language that allow for a large degree of customization (e.g. of calibration targets and routines; see figure 1.). To enable (near-) real-time gaze-dependent displays, the Eyelink parses the eye-movement data on-line, and sends gaze data, as well as other data over the ethernet connection to the display computer. The Eyelink Toolbox allows access to this information via a MATLAB structure. Table 2 lists the main fields of this structure and provides a short description of its contents.

Tables 1 & 2 about here

The use of an interpreted language comes at a cost; native MATLAB code tends to run significantly slower than comparable C code. The penalty of the overhead is about 4 microseconds per MATLAB statement (as reported by the PsychToolbox' SpeedTest.m' routine). Nevertheless, the Eyelink Toolbox implementation, even on a relatively old Power Macintosh with a 400 Mhz G3 processor, is fast enough to enable a near real-time gaze-dependent display. Between eye movement and screen update there is a delay of about 20 ms[1], of which our tests indicate that almost all of it is due to the Eyelink hardware and software on the dedicated Eyelink PC. A millisecond or less of the delay is due to network communication between the Eyelink and display computers. Only a fraction of a millisecond is due to the overhead of using MATLAB in addition to the underlying C-routines. Thus using the Eyelink Toolbox within the MATLAB environment, instead of directly programming in C, adds little cost in terms of execution time. The MATLAB environment, by providing extensive support for numerical processing, in principle also allows the post-processing of eye-movement data within the same environment. At present, support for this additional use of MATLAB is not part of the Eyelink Toolbox.

[1] (We determined the delay from eye-movement to stimulus update to be about 20 ms. This was determined using electro-oculography to independently record eye-movements and a photocell to measure stimulus-update-related changes in light flux on the screen (Cornelissen & Kooijman, to be published).

Example and use

Figure 1 lists an example of a short MATLAB program that uses the Eyelink and Psychophysics Toolboxes to create a simple gaze-dependent display. Step 1 in the example is the initialization of the Eyelink, step 2 the opening of a graphics window using the Psychophysics Toolbox SCREEN routine. In step 3, Eyelink toolbox default values are set and information about the graphics environment is passed onto the Eyelink mex routine and in step 4 the Eyelink's integrated calibration and drift correction procedures are performed. In step 5, data recording commences. In step 6, current gaze position as communicated by the Eyelink is used to show a small dot on the display that moves with the subject's gaze. In case the Eyelink provides no valid gaze data (e.g. during a blink), the screen is immediately blanked. In step 7, graphics window, data file and tracker are closed.

Figure 1 about here

Documentation and availability

The Eyelink Toolbox can be downloaded via the web site http;//psychtoolbox.org/ and, like the Psychophysics toolbox, is available for both the Macintosh and Windows operating systems[1]. Installation consists of adding a single folder (approximately 1 megabyte) to MATLAB's collection of toolboxes. Help is provided via MATLAB's regular conventions. The distribution includes example MATLAB code (such as the program listed in figure 1). The C source code of the toolbox is available (though not for the proprietary subroutines on which the toolbox is based). Note that the Eyelink Toolbox is neither provided nor endorsed nor supported by the manufacturer of the Eyelink Gazetracker. The toolbox may be used freely for teaching or research. It may not be used for commercial gain without permission by the first author of this paper.

Conclusion

The Eyelink Toolbox, in combination with the Psychophysics Toolbox, provides a fast, easy, interactive and powerful means to develop research-grade eye-movement paradigms (e.g. Li, Brenner, Cornelissen & Kim, 2002, Huk, Palmer & Shadlen, 2002).

Acknowledgments

Eyal Reingold and Dave Stampe conceived and developed the Eyelink Gazetracker. We greatly appreciate the effort David Brainard and Denis Pelli have undertaken developing the Psychophysics Toolbox (see http;//psychtoolbox.org/). We thank Eyal Reingold, Francesco Maringelli. Erin Harley and two anonymous referees of an earlier version of this paper for commenting on the manuscript.

References

Brainard, D. H. (1997). The Psychophysics Toolbox. *Spatial Vision*, **10**, 437- 442.

Brenner, E. & Cornelissen, F.W. (2000). Separate simultaneous processing of egocentric and relative positions. *Vision Research,* **40**, 2557–2563

Cornelissen, F. W, & Dobbelsteen, J. (1999). Heading detection with simulated visual field defects. *Visual Impairment Research,* 1999, **1**, 71-84

Cornelissen, F. W. & Kooijman, A. C. The influence of artificial scotomas on eye-movements during visual search. (to be published).

Huk, A. C., Palmer, J. & Shadlen, M. N. (2002). Temporal integration of visual motion information: Evidence from response times. Poster presented at the meeting of the Vision Sciences Society, May 10-15, 2002, Sarasota, Florida, USA.

Li, H-C. O., Brenner, E., Cornelissen, F. W. & Kim, E. S. (2002) Systematic distortion of perceived 2D shape during smooth pursuit eye-movements. (submitted).

Pelli, D. G. (1997). The VideoToolbox software for visual psychophysics. *Spatial Vision*, **10**, 437- 442.

Sheena, D. & Borah, B. (1981). Compensation for some second-order effects to improve eye position measurements. In D.F. Fisher, R.A. Monty, & J.W. Senders (Eds.), *Eye Movements: Cognition and visual perception*. Hillsdale, NJ: Erlbaum.

Stampe, D. M. (1993) Heuristic filtering and reliable calibration methods for video-based pupil-tracking systems. *Behavioral Research Methods, Instruments and computers*, **25**, 137-142.

Tant, M. L. M., Cornelissen, F. W. Kooijman, A. C. & Brouwer, W. H. (2002). Hemianopic visual field defects elicit hemianopic scanning. *Vision Reseach*, **42**, 1339-1348

The MathWorks. (1993). MATLAB User's Guide. The MathWorks, Inc., Natick, MA.

Notes

1. The Eyelink Toolbox is compatible with both the first and second generation of Eyelink gazetrackers.

Figure captions

Figure 1. Listing of a short MATLAB program that uses the Eyelink and Psychophysics Toolboxes to create a simple gaze-dependent display. See main text for details.

Table 1. Listing and description of the main commands available in the Eyelink Toolbox.

Table 2. Listing of the main fields available in the Eyelink data sample structure. Whether these fields are actually filled with useful information depends on the settings of the Eyelink. This can be changed at run-time using the 'command' command of the Eyelink Toolbox (see table 1).

Table 1.

Command	Description of the function
`'Initialize'`	Establishes an ethernet connection between the Eyelink PC and display computer and initializes the Eyelink Gaze tracker.
`'Initializedummy'`	Run in "dummy mode", which allows one to run (and e.g. debug) stimulation programs without a Gaze tracker being connected.
`'Shutdown'`	Close the connection to the Eyelink PC and Gaze tracker.
`'Isconnected'`	Reports the status of the connection (connected, no connection, running in dummy mode).
`'Openfile'`	Opens a file on the Eyelink PC to store data and events in.
`'Closefile'`	Closes the data file.
`'Command'`	Sends a command string to the Eyelink PC (e.g. 'link_sample_data=GAZE' tells the Eyelink to send real-time gaze data to the display computer).
`'Message'`	Sends a message to the Eyelink PC (e.g. "Stimulus On").
`'Initwindow'`	Provide information about the graphics environment (e.g. screen resolution) to the Eyelink Toolbox.
`'Eyeavailable'`	Reports which eye(s) is being tracked.
`'Trackersetup'`	Perform integrated tracker set-up procedure.
`'Dodriftcorrect'`	Perform integrated drift correction procedure.
`'Startrecording'`	Starts recording eye-movement data.
`'Stoprecording'`	Stops recording eye-movement data.
`'Checkrecording'`	Reports whether the Eyelink is recording data.
`'Newsampleavailable'` `'Newfloatsampleavailable'`	Report whether a new data sample (either in int or float format) is available (for real-time purposes).
`'Newestsample'` `'Newestfloatsample'`	Return the most recent data sample (either in int or float format).

Table 2.

Field	Content
time	Time stamp of sample.
flags	Flags to define what data is included in each sample.
gx, gy	Horizontal and vertical gaze position data (x and y) in screen coordinates (pixels) for the left and right eyes (depending on whether they are actually being tracked).
pa	Pupil size or area (depending on settings of Eyelink) of left and right eyes.
rx, ry	Horizontal and vertical resolution of the gaze data (pixels per degree) for the left and right eyes.
hx, hy	Horizontal and vertical head-referenced eye-position data for the left and right eyes.

Figure 1.

```
% Short MATLAB example program that uses the Eyelink and Psychophysics
% Toolboxes to create a real-time gaze-dependent display.

% STEP 1
% Initialization of the connection with the Eyelink Gazetracker.
% exit program if this fails.
if (EYELINK('initialize') ~= 0)
       return;
end;

% STEP 2
% Open a graphics window on the main screen
% using the PsychToolbox's SCREEN function.
screennr = 0; % use main screen
[window, screenRect]=SCREEN(screennr,'OpenWindow', 0);
white=WhiteIndex(window);
black=BlackIndex(window);

% STEP 3
% Provide Eyelink with details about the graphics environment
% and perform some initializations. The information is returned
% in a structure that also contains useful defaults
% and control codes (e.g. tracker state bit and Eyelink key values).
el=initeyelinkdefaults;

% make sure that we get gaze data from the Eyelink
EYELINK('command', 'link_sample_data  = LEFT,RIGHT,GAZE,AREA');
% open file to record data to
EYELINK('openfile', 'demo.edf');

% STEP 4
% Calibrate the eye tracker using the standard calibration routines
EYELINK('trackersetup');
% do a final check of calibration using driftcorrection
EYELINK('dodriftcorrect');

% STEP 5
% start recording eye position
EYELINK('startrecording');
% record a few samples before we actually start displaying
waitsecs(0.1);
% mark zero-plot time in data file
EYELINK('message', 'SYNCTIME');
stopkey=KbName('space');
eye_used = -1;
```

```
% STEP 6
% show gaze-dependent display
while 1  % loop till error or space bar is pressed
       % Check recording status, stop display if error
       error=EYELINK('checkrecording');
       if(error~=0)
              break;
       end
       % check for keyboard press
       [keyIsDown,secs,keyCode] = KbCheck;
       % if spacebar was pressed stop display
       if keyCode(stopkey)
              break;
       end
       % check for presence of a new sample update
       if EYELINK( 'newfloatsampleavailable') > 0
              % get the sample in the form of an event structure
              evt = EYELINK( 'newestfloatsample');
              if eye_used ~= -1   % do we know which eye to use yet?
                     % if we do, get current gaze position from sample
                     x = evt.gx(eye_used+1);    % +1 as we're accessing MATLAB array
                     y = evt.gy(eye_used+1);
                     % do we have valid data and is the pupil visible?
                     if x~=el.MISSING_DATA & y~=el.MISSING_DATA & evt.pa(eye_used+1)>0
                            % if data is valid, draw a circle on the screen at current gaze position
                            % using PsychToolbox's SCREEN function
                            gazeRect=[ x-7 y-7 x+8 y+8];
                            SCREEN(window, 'FrameOval', white,gazeRect,6,6);
                     else
                            % if data is invalid (e.g. during a blink), clear display
                            SCREEN(window, 'FillRect',black);
                     end
              else   % if we don't, first find eye that's being tracked
                     eye_used = EYELINK( 'eyeavailable');  % get eye that's tracked
                     if eye_used == el.BINOCULAR;      % if both eyes are tracked
                            eye_used = el.LEFT_EYE;    % use left eye
                     end
              end
       end % if sample available
end % main loop
% wait a while to record a few more samples
waitsecs(0.1);

% STEP 7
% finish up: stop recording eye-movements,
% close graphics window, close data file and shut down tracker
EYELINK('stoprecording');
SCREEN(window,'close');
EYELINK('closefile');
EYELINK('shutdown');
```