City, University of London Institutional Repository

SOFTWARE METAPAPER

Myex: A Matlab Interface for the Tobii Eyex Eye-Tracker

Pete Richard Jones

UCL Institute of Ophthalmology, 11-43 Bath St, Greater London EC1V 9EL, UK
p.r.jones@ucl.ac.uk

Myex is a Matlab interface for the Tobii EyeX eye-tracker. It allows Matlab users to receive incoming data from the eye-tracker, by providing a data buffer that can receive data from the EyeX, and be queried by the user on demand. Myex enables Matlab users to take advantage of low-cost, portable eye-tracking technology, ideal for use in gaze-contingent psychophysical paradigms, or for users looking to develop assistive devices for individuals with impaired mobility.

(1) Overview
Introduction
The Tobii EyeX (Tobii Technology, Stockholm, Sweden) is a cheap, portable eye-tracker, with integrated head-tracking. It uses near-infrared light reflected from the user's cornea to measure point-of-gaze (2D screen coordinates, in pixels) and eye-ball position (3D spatial coordinates, in millimeters from the screen). The device automatically partials-out head-movements from the gaze estimates, so the user is not required to user a chin or forehead rest.

Unlike most 'research-grade' eye-trackers — which buffer data internally — the EyeX pushes incoming data directly to a user-specified function. This makes it impossible to interface the EyeX directly using Matlab (The MathWorks, Natick, USA), because Matlab is unable to pass invokable function-pointers to external applications.

The present software solves this problem using an intermediate C subroutine ("myex.mex"), which is capable of receiving/buffering data from the Tobii EyeX on the one hand, and of passing the data on demand to Matlab on the other.

Implementation and architecture
The software was written primarily in C, and can be compiled into a Mex file for use with Matlab. Example compilations are provided for windows 32-bit and 64-bit. The distribution additionally includes a Minimal Working Example written in Matlab.

Use has been intentionally kept extremely simple. There are no dependencies on 3rd party Matlab toolboxes, and the code provides just three commands: myex('connect'), myex('getdata'), and myex('disconnect'). A full Minimal Working Example is given in **Listing 1**. This code is also included as part of the distribution.

Only one connection is permitted at a time, and "disconnect" should be used to close the connection to the eye-tracker after use. Matlab's "clear all" command will also force any open connection to close, and can be used defensively prior to the use of "connect".

The "getdata" command returns a matrix containing any/all additional data since the previous call (one row per sample). Thus, a call to "getdata" every 500 milliseconds would be expected to return a matrix with approximately 25 rows of data (i.e., given the hardware's sampling rate of ~50 Hz). Note, however, that the eye-tracker only pushes data when at least one eye is being tracked, so fewer rows (samples) than expected may be returned. If no new data are available, an empty matrix is returned. Each row (sample) of data contains 12 columns, as detailed in **Table 1**.

Quality control
Myex has been used continuously within our lab since 2016, and has been used successfully for several gaze-contingent psychophysical experiments. Furthermore, manual end-to-end testing has been carried out to ensure stability across different machines and Matlab versions. Note that the code does not in any way modify or affect the data returned by the Tobii EyeX Engine, and Myex introduces negligible additional processing overheads and/or lag. (NB: the latency of any gaze estimate is determined primarily by any smoothing applied within the Tobii EyeX engine itself, and by the transmission speed of the USB connection: <10 ms.).

(2) Availability
Operating system
Myex is compatible with Windows 7 and Windows 10 (the only platforms supported by the EyeX eye-tracker).

```matlab
1   % connect to EyeX Engine
    myex('connect')

    % open tracking window, set axes to screen size
5   hFig = figure();
    hold on
    hTxt = text(0,0,'waiting for data');
    hDat = plot(NaN,NaN,'+', 'MarkerSize',14);
    set(gca, 'xlim',[0 1920], 'ylim',[0 1080], 'Ydir','reverse'); % fix ...
        resolution, since Matlab's ability to detect current resolution is poor.
        at best
10  hold off

    % clear any data in buffer
    myex('getdata');

15  % allow to track until window closed
    x_all = [];
    while ishandle(hFig) % check if figure still open
        % get data
        x = myex('getdata');
20
        % if data returned
        if ~isempty(x)
            % determine eyeball distance
            z_mm = x(end,[8 11]);
25          isvalid = x(end,[4 5])==1;
            isvalid = isvalid & (z_mm>0.001); % defensive (shouldn't be ...
                necessary)
            z_mm = z_mm(isvalid);
            z_mm = nanmean(z_mm);

30          % update plot
            set(hTxt, 'String', sprintf('Distance = %1.2f cm', z_mm/10));
            set(hDat, 'xdata',x(end,1), 'ydata',x(end,2));
            set(gca, 'XLim',[0 max([x(end,1) get(gca,'XLim')])], 'YLim',[0 max..
                ([x(end,2) get(gca,'YLim')])] );
            drawnow();
35
            % add data to store
            x_all = [x_all; x]; %#ok<AGROW> This is innefficient memory-...
                allocation, but ok for present purposes
        else
            set(hTxt, 'String', 'Waiting for data');
40      end
        pause(1/50); % run at 50 Hz
    end

    % disconnect from EyeX Engine
45  myex('disconnect')
```

Listing 1: Minimal Working Example demonstrating how the code can be used to interface with the EyeX hardware (see body text for details).

Programming language
Myex is compatible with all known versions of MATLAB.

Additional system requirements
Myex is designed to interface with the Tobii EyeX eye-tracker (Tobii Technology, Stockholm, Sweden), which requires a USB 3.0 connection.

Dependencies
Myex requires is compatible with all versions of the Tobii EyeX Interaction Engine from v1.2.0 onwards (at the time of writing the latest version is v1.9.4). There are no MATLAB dependencies. However, users wishing to compile Myex from source may need to install an appropriate C/C++ compiler (run "mex -setup" from within MATLAB for more info.

Table 1: Data fields returned for each sample. Note that gaze and eyeball location are estimated separately, and the timestamps for each may differ slightly. For most applications, however, they can be treated as identical.

id	Field name	Description
1	X (px)	Horizontal gaze location on the screen
2	Y (px)	Vertical gaze location on the screen
3	EyeGazeTimestamp (µs)	Timestamp for gaze estimate
4	HasLeftEyePosition (0 or 1)	Whether left eye was detected
5	HasRightEyePosition (0 or 1)	Whether right eye was detected
6	LeftEyeX (mm)	Left eyeball location in physical space
7	LeftEyeY (mm)	"
8	LeftEyeZ (mm)	"
9	RightEyeX (mm)	Right eyeball location in physical space
10	RightEyeY (mm)	"
11	RightEyeZ (mm)	"
12	EyePosTimestamp (µs)	Timestamp for location estimates

List of contributors

Pete Jones wrote the software and is its current maintainer.

Archive and Code Depository

GitHub
Name: Myex
Persistent identifier: https://doi.org/10.5281/zenodo.998562
URL: https://github.com/petejonze/myex
Licence: GNU GPL v3.0
Current version: v1.0.0
Publisher: Zenodo
Date published: 28/09/2017

(3) Reuse potential

Myex is suitable for any MATLAB users looking to take advantage of low-cost, portable eye-tracking technology. The relatively low sampling rate of the eye-tracker (~50 Hz) makes it inappropriate for eye-movement researchers looking to perform detailed temporal analyses. However, its simplicity and robustness to head-movements makes it ideal for users looking to add-contingent functionality to an application. For example, it could be used to present stimuli to specific regions of the retina [1, 2], or as an input mechanism for users looking to develop assistive devices for individuals with impaired mobility [3]. The ability of the eye-tracker to track monocularly is also particularly attractive for working with patients, who may not always have two functioning eyes. In our lab we are currently validating its use both as a stimulus-positioning and an input-response mechanism, as part of a low-cost visual-impairment screening device targeted at developing countries.

The general solution detailed here could also be extended to programming languages other than MATLAB. Note, however, that many popular languages such as Python, C#, or C++, allow users to supply invokable function-pointers directly to external applications. They therefore do not require data to be buffered externally by an intermediate application such as Myex, and can instead interface with the Tobii EyeX engine directly.

Acknowledgements

Competing Interests

The author has no competing interests to declare.

References

1. **Jones, P R, Kalwarowsky, S, Atkinson, J, Braddick, O J** and **Nardini, M** 2014 Automated measurement of resolution acuity in infants using remote eye-tracking. *Invest. Ophthalmol. Vis. Sci.*, 55: 8102–8110. DOI: https://doi.org/10.1167/iovs.14-15108
2. **Murray, I C, Fleck, B W, Brash, H M, MacRae, M E, Tan, L L** and **Minns, R A** 2009 Feasibility of saccadic vector optokinetic perimetry: A method of automated static perimetry for children using eye tracking. *Ophthalmology*, 116: 2017–2026. DOI: https://doi.org/10.1016/j.ophtha.2009.03.015
3. **Jacob, R J K** and **Karn, K S** 2003 in *Mind*, Hyona, J (ed.). 2: 573–605 (Elsevier Science BV, 2003).