UserGuide

Tobii®Toolbox for Matlab

Product Description & User Guide

tobii
www.tobii.com

Tobii Toolbox for Matlab

Product Description and User Guide

Version 1.1, released on 02.07.2010.

Original release of the Tobii Toolbox, Version 1.0, released on 13.11.2009,

by Andy Shaw, Tim Holmes & Johannes M. Zanker.

Contents

1 Tobii Toolbox for Matlab 4

1.1 Before installation 4

2 Installation & Deployment 5

2.1 MATLAB setup 5

3 Visual C++ Module (EyeTracker.exe) 6

3.1 Tobii Eye Tracking dialog box 7

3.1.1 Calibrate box 7

3.1.2 Track status box 8

4 Matlab – SDK Functions 9

4.1 ConnectTo(serverAddress) 9

4.2 Checking the trackability of the subject (ITetTrackStatus Interface) 9

4.2.1 TrackStatus 9

4.2.2 EndTrackStatus 10

4.3 Calibration (ITetCalibProc & ITetCalibPlot interfaces) 10

4.3.1 Calibrate(numPoints) 10

4.3.2 ReCalibration 11

4.3.3 ClearPlot 11

4.4 Collecting gaze data (ITetClient & ITetClientEvents interfaces) 11

4.4.1 TrackStart(fixation, filename) 11

4.4.2 TrackStop 12

4.5 Synchronizing events with gaze data (ITetClient6) 12

4.5.1 GetTetTime 13

5 Gaze Data Format 14

5.1 Eye tracker gaze data quick reference 14

5.2 Validity codes 14

6 System Diagram 16

6.1 High level diagram 16

6.2 Function diagram 16

7 Tobii Toolbox demo script 17

7.1 ToolboxDemo script 17

1 Tobii Toolbox for Matlab

The Tobii Toolbox provides a basic interface between Matlab and the Tobii TX Series Eye Trackers (via the Tobii Software Development Kit) in a Windows environment. As such it enables the user to run eye tracking experiments directly from Matlab and to take full advantage of the stimulus generation and data analysis power that Matlab, and associated toolboxes, provide.

1.1 Before installation

To use the Tobii Toolbox for Matlab, your computer must fulfill the following software requirements:

- Tobii TX Series Eye Tracker, with firmware version 1.1.19 or higher
- Matlab, versions R2007b - R2010a
- Tobii Software Development Kit (SDK), versions 2.4.12 or higher
- Microsoft Windows, 32-bit editions (Vista, XP or 7)

Your computer must also comply with the Matlab hardware requirements specified at http://www.mathworks.com/support/sysreq/current_release/index.html.

Important notice: Do not install the Tobii Toolbox for Matlab on the same computer as Tobii Studio. This issue will be fixed in future releases of Tobii Studio. Read the release notes of future Tobii Studio releases for more information.

2 Installation & Deployment

The Tobii Toolbox is easy to install and deploy. All the files required are contained in a single ZIP folder.

To install the Tobii Toolbox:

1. Connect the USB memory stick to the computer where you installed Matlab.
2. In the AutoPlay dialog box, click "Open folder to view files in Windows Explorer".
3. In Windows Explorer, double-click the Tobii Toolbox v1.X.zip folder.
4. Select all the files in the ZIP folder and copy them to a folder on your computer.

The following files should be copied from the ZIP folder:

- ConnectTo.m
- TrackStatus.m
- EndTrackStatus.m
- Calibrate.m
- ReCalibration.m
- ClearPlot.m
- TrackStart.m
- TrackStop.m
- GetTetTime.m
- ToolboxDemo.m
- TobiiLogo.bmp
- EyeTracker.exe (Visual C++ module)[1]

[1] Because this module is compiled as an executable file, no Visual C++ software is required to run it.

2.1 Matlab setup

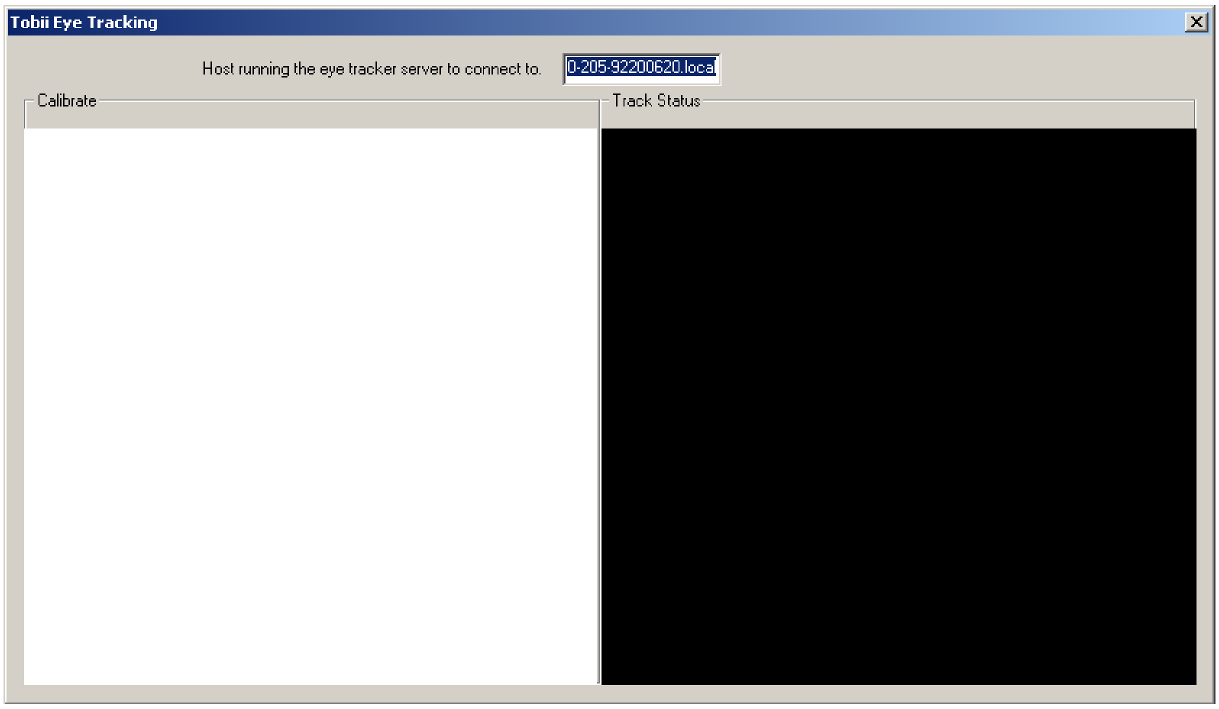
To access the individual files, you first need to link the folder, where the Tobii Toolbox files were extracted to, to Matlab.

To link the file folder to Matlab:

1. Open Matlab.
2. Click the File drop-down menu and select Set Path…. The Set Path dialog box opens.
3. Click Add Folder; a Windows Explorer dialog box opens.
4. In the Windows Explorer dialog box, browse to the folder containing the toolbox files. Click the folder to select it and then click Open to add it to the Set Path dialog box.
5. In the Set Path dialog box, click Save and then Close to complete the task.

3 Visual C++ Module (EyeTracker.exe)

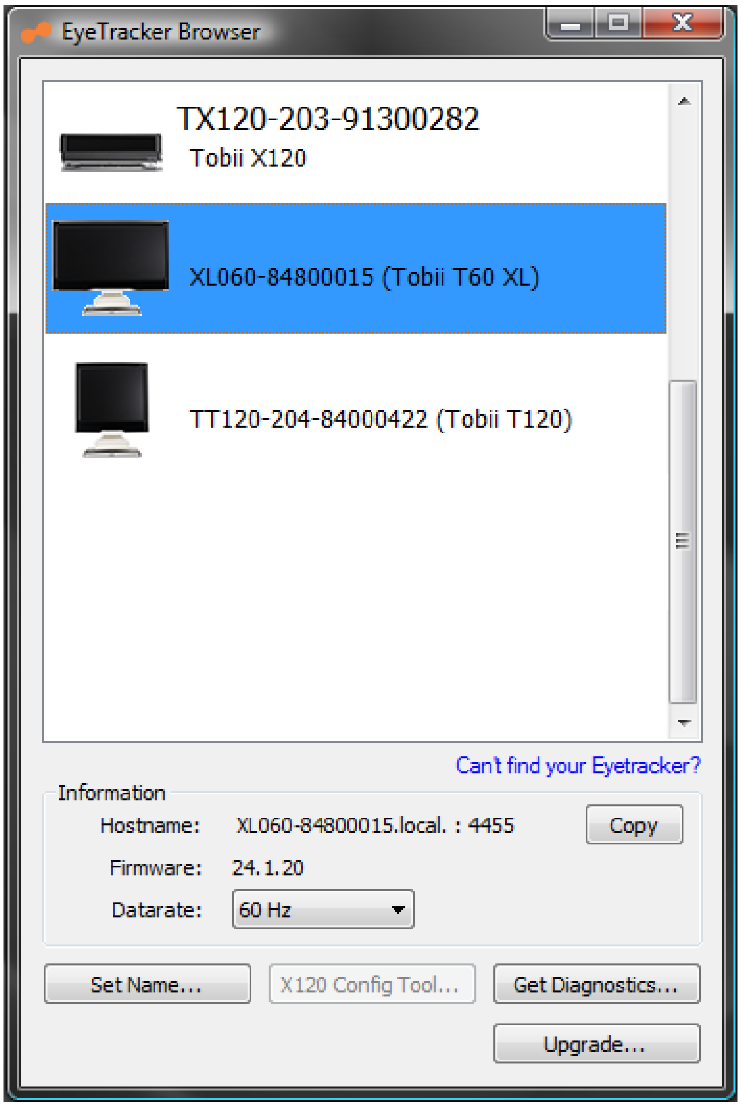
The Visual C++ module – EyeTracker.exe – is responsible for communications between Matlab and the Tobii Eye Tracker via the Tobii SDK. This module contains a user graphical interface that displays the current eye tracking status as well as the calibration results. To run the EyeTracker.exe module in Matlab, use the ConnectTo(server_Address) Matlab function.



The figure above shows the Tobii Eye Tracking dialog box launched by the call function ConnectTo(server_Address) in Matlab. The following TET server address was used as an input to the function: TT120-205-92200620.local.

To find the address of the Tobii Eye Tracker server, you first need to open the Tobii EyeTracker Browser on your computer:

1. Click the Windows Start Menu, and select All Programs
2. Select Tobii and then click Tobii EyeTracker Browser. The Tobii EyeTracker Browser dialog opens.



Information

Hostname: XL060-84800015.local. : 4455

Firmware: 24.1.20

Datarate: 60 Hz

3. Select the eye tracker you wish to connect to by clicking the eye tracker in the eye tracker list box. The server address (Hostname) of the eye tracker is displayed in the Information box. This name is similar to the serial number located at the back or bottom of the eye tracker.
4. Click Copy to copy the eye tracker Hostname. Close or minimize the EyeTracker Browser dialog box.
5. Change to Matlab and use the paste keyboard shortcut to paste the eye tracker hostname into the ConnectTo function.

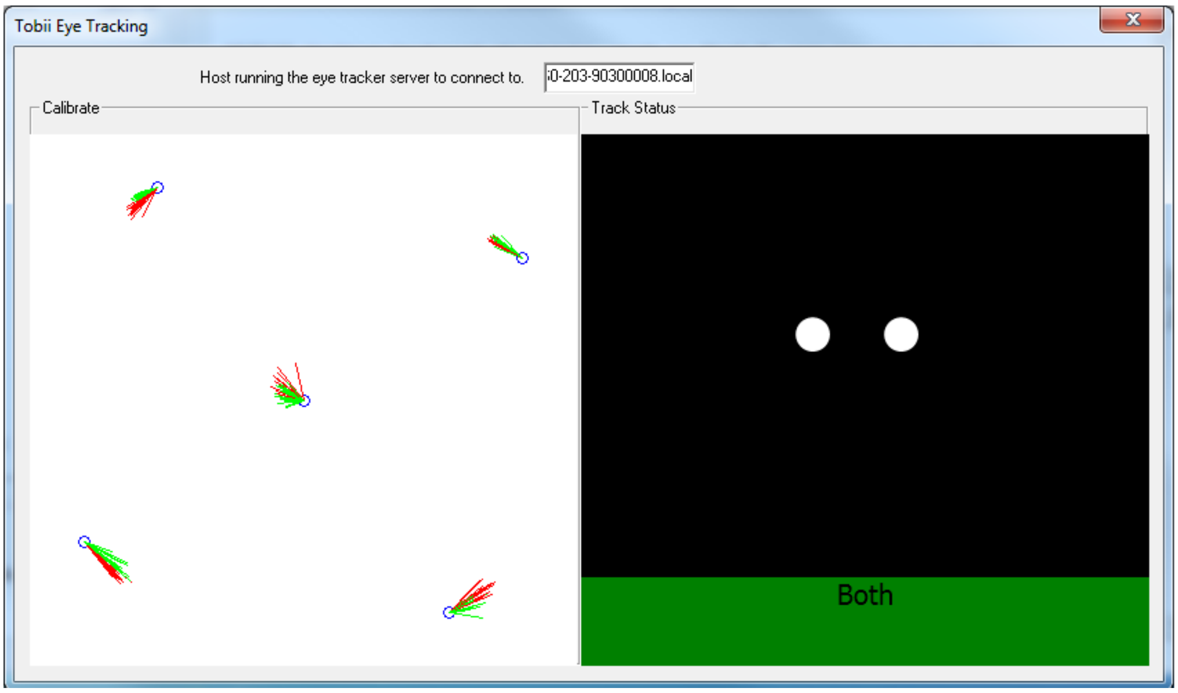
The Tobii EyeTracker Browser is installed automatically when you install the Tobii SDK on the computer.

3.1 Tobii Eye Tracking dialog box

The Tobii Eye Tracking dialog box displays the current eye tracking status, calibration results as well as the Tobii Eye Tracker server address entered in the ConnectTo function (Host running the eye tracker server…).

3.1.1 Calibrate box

The white ActiveX control box on the left hand side of the Tobii Eye Tracking dialog box provides feedback on calibration quality. This box is updated after each calibration is performed using the Calibrate function in the Tobii Toolbox (see Section 4.2 for more information).



The calibration plot shows error vectors – green and red lines (the size of each green line indicates the difference between the gaze point calculated by the eye tracker and the actual dot position). Long lines or absence of lines indicate a poor calibration, whereas short lines indicate a good calibration. The red lines represent the left eye results, while the green lines display the results for the right eye.

3.1.2 Track status box

The black ActiveX control box on the right hand side of the Tobii Eye Tracking dialog box provides feedback on the ability of the Eye Tracker to track the subject. This is activated using the TrackStatus function in the Tobii Toolbox (see Section 4.1 for more information).

Within the black colored area, the position of the subject being tracked is shown by two dots representing the eyes. The text bar shows the tracking ability. The bar uses both colors and text to show if one or both eyes are found. The color bar is dark green with the text “Both” if both eyes were found. If one eye is found, the color is orange, yellow or bright green and text is “Left”, “Right” or “Unknown” depending on how sure the system is that it detected the correct eye and which eye it detected. If no eyes were found, the color bar is red and text displayed is “Not found”. Good tracking ability is thus shown by a dark green bar, while a red or flickering bar means that tracking ability is low.

4 Matlab – SDK Functions

The functions included in this toolbox are contained in the Matlab files (*.m), and have the same names as the files that contains them. To run these functions, enter the name of the function in the Matlab Command Window along with any arguments that need to be added. These functions then trigger the functionality provided by the Tobii SDK via the EyeTracker.exe Visual C++module, with the exception of GetTetTime, which calls the Tobii SDK directly from Matlab (see Section 4.4). The Tobii SDK has a number of different interfaces, each containing methods and properties enabling you to use the functionality of the eye tracker. This section describes the different SDK interfaces implemented in this toolbox and explains which methods are used by the MATLAB functions.

4.1 ConnectTo(serverAddress)

The ConnectTo(server_Address) Matlab function is used to deploy the EyeTracker.exe module within Matlab.

ConnectTo

Connects Matlab to the Tobii Eye Tracker server via the Tobii SDK.

Syntax

ConnectTo(*server_address*)

Description

This function runs the VC++ executable EyeTracker.exe and sets the server address for the eye tracker.

server_address is a string defining the name or address of the server on which the eye tracker is running.

When the function is executed, the Tobii Eye Tracking dialog box opens, displaying the address of the Tobii Eye Tracker server (Host running the eye tracker server to connect to), together with the Calibrate and Track Status boxes (see Chapter 3 for more information).

4.2 Checking the trackability of the subject (ITetTrackStatus Interface)

This interface provides a tool that displays the subject's eyes (as two white circles) in real time. This tool allows you to verify that the subject is positioned correctly and that both eyes are being picked up. The following Matlab functions are provided to utilize this interface.

4.2.1 TrackStatus

TrackStatus

Triggers real-time display of a graphical interface to validate that eyes are being successfully detected.

Syntax

TrackStatus

Description

This function calls the Connect method to connect to the ITetTrackStatus interface. It then calls the Start method to start feeding back the position of the eyes of the subject. This is done through a graphical interface where the eyes are shown as two white dots on a black background. This is also a color-coded bar at the bottom containing text to show if both, neither or only one of the eyes are detected.

4.2.2 EndTrackStatus

EndTrackStatus

Terminates real-time display of the graphical interface to validate that eyes are successfully being detected.

Syntax

EndTrackStatus

Description

This function calls the Stop method to stop updating the graphical display.

4.3 Calibration (ITetCalibProc & ITetCalibPlot interfaces)

These interfaces provide the functionality needed to calibrate the eye tracker. The ITetCalibProc interface is used to perform the calibration of the eye tracking system for each subject being tracked. The ITetCalibPlot interface then provides feedback enabling you to inspect the quality of the calibration. This is done through a graphical interface showing the calibration area.

Each calibration point is shown as a circle and the samples for each point are shown as two lines (green for the right eye, and red for the left eye) from the calibration point to the calculated gaze point of the subject. Long lines indicate a poor calibration while short lines indicate a good calibration.

The following Matlab functions are provided to perform a calibration.

4.3.1 Calibrate(numPoints)

Calibrate

Triggers calibration of the participant.

Syntax

Calibrate(*num_points*)

Description

This function calls the Connect method to connect to the ITetCalibProc interface. It also calls the Connect method for the ITetCalibPlot interface. The StartCalibration method is then called. This brings up the full screen calibration window and displays each of the calibration points. The number of calibration points is defined by the argument numPoints passed into the Matlab function as an integer.

num_points is numeric and corresponds to the number of points to be used in the calibration. It should be set to 2, 5 or 9.

Once the calibration has completed, the SetData and UpdateData methods provided by the ITetCalibPlot interface are called. This sets the data received by the calibration and updates the feedback in the graphical interface.

4.3.2 ReCalibration

ReCalibration

Triggers a re-calibration of all unsuccessfully calibrated points.

Syntax

ReCalibration

Description

Once a calibration has been performed, you can see the quality of the calibration from the feedback provided by the user interface. If you are not satisfied with the calibration then you can call this function to run through the calibration again.

Note: The ReCalibration function is currently set to merely recalibrate the points for which it currently has no calibration data – in other words, for a 5-point Calibration with only 1 point calibrated, the ReCalibration should attempt to calibrate the missing 4 points.

4.3.3 ClearPlot

ClearPlot

Removes calibration plot from the screen.

Syntax

ClearPlot

Description

This function can be used to clear the feedback from the graphical interface once you are satisfied with the calibration. It calls the ClearData method provided by the ITetCalibPlot interface.

4.4 Collecting gaze data (ITetClient & ITetClientEvents interfaces)

The ITetClient interface enables you to receive the gaze data produced by the Tobii Eye Tracker. This data is exposed through regularly fired events controlled using the ITetClientEvents interface. The following Matlab functions are provided to start and stop tracking, and for collecting the gaze data produced.

4.4.1 TrackStart(fixation, filename)

TrackStart

Starts the capture of eye movement data from the eye tracker.

Syntax

TrackStart(*fixation_ind, trackfile*)

Description

fixation_ind is an integer with a value of 0 or 1 and is used to toggle the display of fixation location on the screen during the eye-tracking session (useful during experimental design).

> *fixation_ind = 1* causes the current fixation location to be displayed on screen
>
> *fixation_ind = 0* inhibits the display of the current fixation location

trackfile is a string containing the filename to be used for the eye-tracking data for the current tracking session. If no path is specified, this will be the current directory in Matlab.

This function calls the Connect method to connect to the ITetClient interface. The StartTracking method is then called to start collecting any gaze data events that are fired. The fixation parameter passed into this function turns a fixation square on or off. When turned on this square appears on the screen following the gaze of the subject. It appears as green if the eyes are being picked up properly, turning to red if the eyes are lost at any point. This input argument is an integer and should be 0 if you want the fixation square turned off, and 1 if you want it turned on. When tracking every time new gaze data is available, the OnGazeData event is fired through the ITetClientEvents interface. The gaze data is then added to the file defined by the filename argument passed in to this function as a string. The format of the gaze data file produced is described in the next section. To read this data into Matlab, use the *csvread* command.

4.4.2 TrackStop

TrackStop

Ends the capture of eye-movement data from the eye tracker.

Syntax

TrackStop

Description

This function calls the StopTracking method through the ITetClient interface to end a tracking session and stop the collection of gaze data. The file containing the eye-tracking data is stored as a *.csv* file and can now be loaded into Matlab for analysis using the *csvread* command.

4.5 Synchronizing events with gaze data (ITetClient6)

When using the Tobii Toolbox for Matlab, Matlab determines when certain events should be initiated (for example, when gaze tracking should be initiated), while the Tobii Eye Tracker server collects the raw gaze data values and sends them to the computer running Matlab. The gaze data is originally time-stamped on the Tobii Eye Tracker server using a high resolution counter located in the eye tracker hardware. The Matlab events on the other hand, are handled in the Matlab computer and are likely to be time-stamped using the computer's hardware clock or counter. To synchronize data from the two different sources, the Tobii SDK installation contains a Tobii proprietary communications protocol called TTime that runs as a background thread on both the Tobii Eye Tracker server and the Matlab computer. This protocol compensates for the time drift between the counters of the two hosts, and makes sure that both the Matlab events and the gaze data are time-stamped with the same time reference. The following function enables Matlab to access the TTime protocol.

4.5.1 GetTetTime

GetTetTime

Obtains the current time stamp from the Tobii eye tracker.

Syntax

t = GetTetTime

Description

This function is different from the other functions in the toolbox since it calls the Tobii SDK (ITetClient6) directly from Matlab, rather than via the C++ interface. There are no inputs to the function and it simply returns the TTime in microseconds as a long integer, allowing you to obtain the TTime for any Matlab- initiated event. It is important to understand that this time stamp is from the eye tracker and thus relates directly to the two time stamp fields at the start of each row of data in the gaze data (*.csv*) file specified with the *TrackStart* command. This allows you to calculate the offset between the time stamp of an event initiated in Matlab, such as display of stimulus, and the eye-movement data recorded by the eye tracker.

5 Gaze Data Format

The gaze data produced is saved as a comma-separated values (.csv) file. Each line in the file represents each new gaze data event fired during a tracking session. The name of the file is defined by the user and passed to the TrackStart function (see Section 4.3.1). Each line of the gaze data consists of sixteen data fields.

5.1 Eye tracker gaze data quick reference

This table should be used for quick reference. For more information about eye tracker gaze data, see Appendix A of The Tobii SDK User Manual (pages 132–134).

Field	Type	Description
Time stamp (second)	4 bytes signed integer	Time stamp when gaze data was sampled.
Time stamp (microsecond)	4 bytes signed integer	Microsecond fraction of time stamp when gaze data was sampled.
Left eye horizontal gaze target position	4 bytes floating point number	Gaze position related to current calibration. Increases at subjects' right.
Left eye vertical gaze target position	4 bytes floating point number	Gaze position related to current calibration. Normally increases downwards.
Left eye horizontal position as seen by the eye tracker.	4 bytes floating point number	The eye position as registered by the eye tracker. 0 is leftmost and 1 is rightmost.
Left eye vertical position as seen by the eye tracker.	4 bytes floating point number	The eye position as registered by the eye tracker. 0 is topmost and 1 is bottommost.
Left eye distance (millimeter)	4 bytes floating point number	Distance between the subjects' eye and the eye tracker.
Left eye pupil size (millimeter)	4 bytes floating point number	The length of the longest chord of the pupil ellipse.
Left eye validity code	4 bytes signed integer	An estimate of how valid all left eye data are.
Right eye horizontal gaze target position	4 bytes floating point number	Gaze position related to current calibration. Increases at subjects' right.
Right eye vertical gaze target position	4 bytes floating point number	Gaze position related to current calibration. Normally increases downwards.
Right eye horizontal position as seen by the eye tracker.	4 bytes floating point number	The eye position as registered by the eye tracker. 0 is leftmost and 1 is rightmost.
Right eye vertical position as seen by the eye tracker.	4 bytes floating point number	The eye position as registered by the eye tracker. 0 is topmost and 1 is bottommost.
Right eye distance (millimeter)	4 bytes floating point number	Distance between the subjects' eye and the eye tracker.
Right eye pupil size (millimeter)	4 bytes floating point number	The length of the longest chord of the pupil ellipse.
Right eye validity code	4 bytes signed integer	An estimate of how valid all right eye data are.

Important note: The two time stamp values (seconds and microseconds) should be combined into a single value, where the "Time stamp (microsecond)" value is added as the microsecond fraction of the "Time stamp (seconds)"value.

5.2 Validity codes

These codes are based on whether the system is able to correctly identify both eyes. It is recommended that the validity codes always be used for data filtering to remove data points that are obviously incorrect.

The validity code takes one of five values for each eye ranging from 0 to 4, with the following interpretation:

- 0 – The eye tracker is certain that the data for this eye is correct. There is no risk of confusing data from the other eye.
- 1 – The eye tracker has only recorded one eye, and has made some assumptions and estimations regarding which is the left and which is the right eye. However, it is still very likely that the assumptions made are correct. The validity code for the other eye is in this case always set to 3.
- 2 – The eye tracker has only recorded one eye, and has no way of determining which is the left eye and which is the right eye. The validity code for both eyes is set to 2.

- 3 – The eye tracker is fairly confident that the actual gaze data belongs to the other eye. The other eye will always have validity code 1.
- 4 – The gaze data is missing or definitely belongs to the other eye.

Possible combinations:

Left eye validity code	Right eye validity code	Eyes detected	Eye identification
0	0	Both	Correctly identified
4	0	Right	Correctly identified
0	4	Left	Correctly identified
3	1	Right	Estimated as probable
1	3	Left	Estimated as probable
2	2	One eye	Uncertain
4	4	None	Uncertain

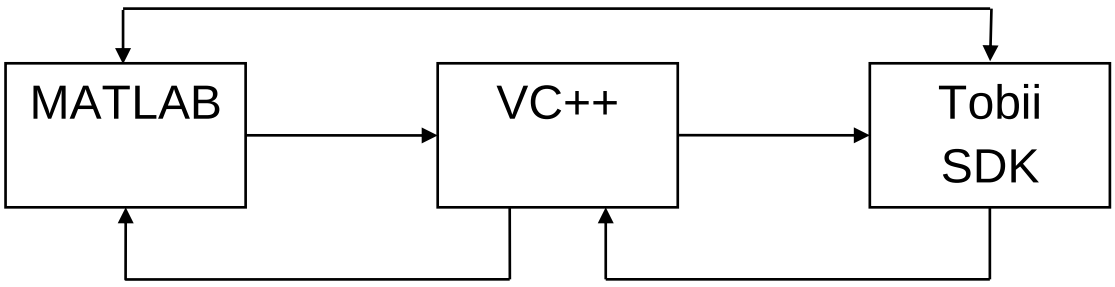
Validity codes should be used for data filtering to remove data points that are obviously incorrect. If you export the raw data file, we recommend removing all data points with a validity code of 2 or higher.

6 System Diagram

The following diagrams show how the different parts of the system interact with each other.

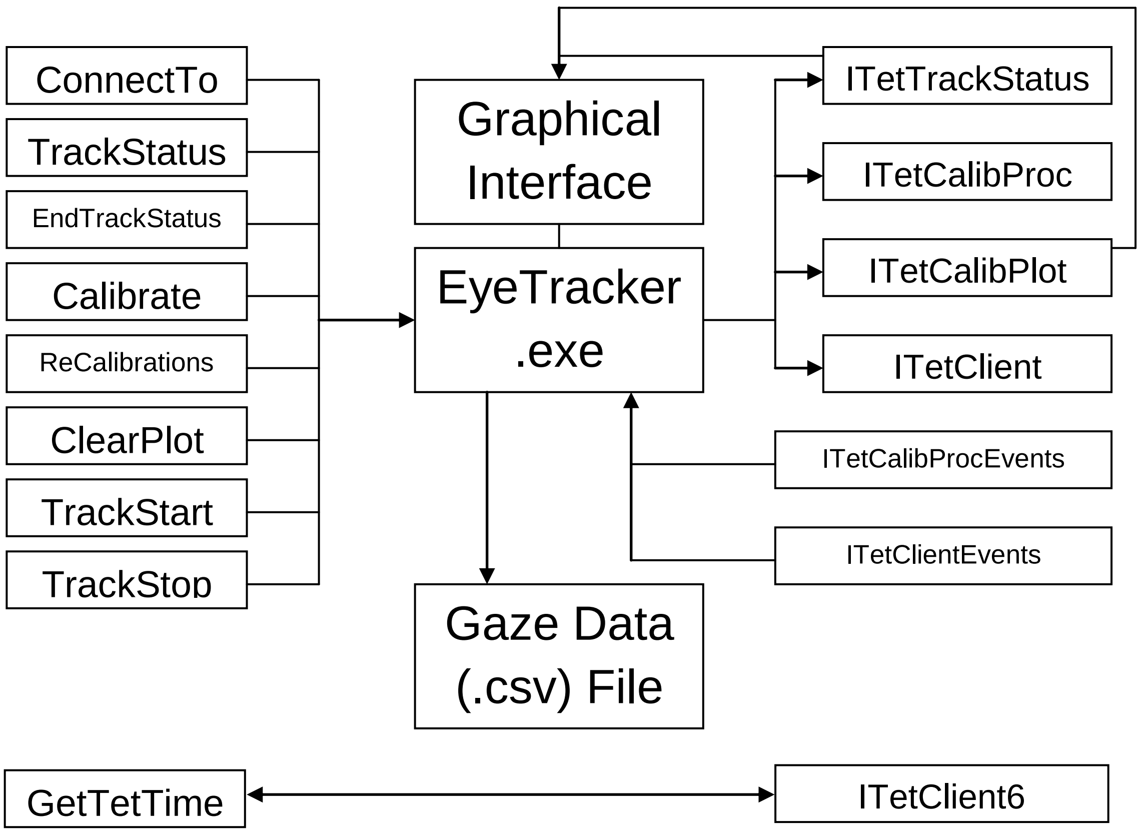
6.1 High level diagram

This diagram shows the interaction between MATLAB, the Visual C++ module and the interfaces of the Tobii SDK at a very basic, high level.



6.2 Function diagram

This diagram shows the MATLAB functions provided with the toolbox, and the Tobii SDK interfaces used.



7 Tobii Toolbox demo script

While the previous sections of this document outline in detail, the functionality in the Tobii Toolbox for Matlab and how to call each module, it is often easier for programmers to see the calls being used in a script. For this reason, ToolboxDemo.m is a simple demonstration program that has been included with the toolbox. It is self-contained, requiring only the TobiiLogo.bmp image file as input.

ToolboxDemo.m includes examples of calls to all functions in the Tobii Toolbox, and is commented throughout making it easy to follow. After connecting to the eye tracker, it will perform a calibration before displaying an image, capturing eye-movement data and producing a basic fixation plot. As such, it can be used as a skeleton for a simple eye-tracking experiment, but it should be noted that the simplest method of displaying an image from Matlab is used, and it most likely that you will want to explore other methods in Matlab or use existing, freely available toolboxes such as Psychtoolbox or Cogent.

To illustrate how the raw data from the Tobii Eye Tracker might be used, a very basic fixation routine is included. This is intended for illustration purposes only and it is anticipated that programmers will either develop or use established scripts for data analysis.

To run the demo:

1. Ensure that the directory containing the Toolbox files is added to the Matlab path (see Section 2.1).
2. Enter the command ToolboxDemo in the command window: >> ToolboxDemo.

7.1 ToolboxDemo script

```
% ************************************************************************
%
% ToolboxDemo.m
%
% Author: Tim Holmes
% Institution: Royal Holloway, University of London
% Date: July 2nd, 2010
% Version: 1.0
%
% Demonstration script containing examples of all calls to the Tobii
% Toolbox v1.1 - please note, this same script WILL work with v1.0, with
% the GetTetTime call (see in line comments).
%
% Before running this script it is essential that the installation
% instructions in the Tobii Toolbox User Documentation v1.1 have been
% completed.  In particular, the Tobii Toolbox must be unzipped to a
% directory and that path included in the Matlab paths and the Tobii SDK
% v2.4.12 or higher must be installed on the same machine as Matlab.
%
% This script is structured as follows:
%
% 1. Initialisation and connection to the Tobii Eye-tracker
% 2. Calibration of a participant
% 3. Display of a stimulus
% 4. Initiate eye-tracking
% 5. Get raw eye-movement data from Tobii Eye-tracker
% 6. Produce crude fixation plot
%
% ************************************************************************

warning off all;

% ************************************************************************
%
% 1. Initialisation and connection to the Tobii Eye-tracker
%
% ************************************************************************

fixationduration = 6; %100/(1000/120)  i.e. 100ms window/duration of a single row on the 120Hz
file, use 6 for 60Hz (binocular)
```

```
fixationhrange = 0.029; %+/- 0.5 degree as a percentage of the screen size (standard T120
monitor with 57cm viewing distance)
fixationvrange = 0.037; %+/- 0.5 degree as a percentage of the screen size (standard T120
monitor with 57cm viewing distance)
fixation(1,1:6) =0;

%get path info
path = input('enter path for input/output files (easiest to use Tobii Toolbox path for demo):
','s');

%Call to Tobii Toolbox to connect to eye-tracker
tobiiserver = input('Tobii server name? e.g. TT120-205-92200620.local: ','s');
ConnectTo(tobiiserver);

% ***************************************************************************
%
% 2. Calibration of a participant
%
% ***************************************************************************

calibrationpoints = input('How many calibration points? (2,5 or 9): ','s');
TrackStatus; %Call to Tobii Toolbox to display the SDK GUI to confirm detection of both eyes
trackstatus = input('Ready to calibrate?: ','s');
while trackstatus == 'N' || trackstatus == 'N'
    trackstatus = input('Ready to calibrate?: ','s');
end;
EndTrackStatus; %Call to Tobii Toolbox to clear the GUI
Calibrate(str2double(calibrationpoints)); %Call to Tobii Toolbox to perform calibration
calibrated = input('Recalibration needed (Y/N)?: ','s');
while calibrated == 'Y' || calibrated == 'y'
    ReCalibration; %Call to Tobii Toolbox to perform recalibration of unsuccesful points
    calibrated = input('Recalibration needed?: ','s');
end;
ClearPlot; %Call to Tobii Toolbox to remove the calibration results plot from the screen

% ***************************************************************************
%
% 3. Display of a stimulus
%
% For the demo this simply reads and display a bitmap of the Tobii logo.
% Any method for generation and display of stimuli availble to Matlab could
% be inserted here, for example using Psychtoolbox or Cogent
%
% ***************************************************************************

A=imread(strcat(path,'\TobiiLogo.bmp'));
imagesc(A);
axis off;
axis image;
set(1,'MenuBar','none');
a=get(1,'Position');
set(1,'Position',[40 40 40+(2*a(3)) 40+(2*a(4))]);

% ***************************************************************************
%
% 4. Initiate eye-tracking
%
% ***************************************************************************

starttime = GetTetTime; %Call to Tobii Toolbox to capture the TTime (timestamp from the Tobii
Eye-tracker)

%Start the eyetracking
trackfile = strcat(path,'\tobiidata.csv');
TrackStart(0,trackfile); %Call to Tobii Toolbox to commence tracking without displaying the
fixation location on screen and return raw data in trackfile
pause(5); %Track gaze for 5 seconds
TrackStop; %Call to Tobii Toolbox to end tracking


% ***************************************************************************
%
% 5. Get raw eye-movement data from Tobii Eye-tracker
%
% ***************************************************************************

DATA = csvread(trackfile);
```

```
% ***************************************************************************
%
% 6. Produce crude fixation plot
%
% Now you have the raw data from the eye-tracker in Matlab matrix DATA, you
% can proceed with any kind of analysis you wish.  The following code is a
% fairly unsophisticated fixation point analysis and plot and is included
% merely for illustration of how you might proceed.  Details of the
% structure and format of entries in DATA are given in Appendix A of the
% Tobii SDK Developer's Guide.
%
% ***************************************************************************


trackstarttime = DATA(1,8)+(DATA(1,9)/1000000);

%calculate the average x and y gaze coordinates from the left and right eye
%positions from Tobii
DATA(:,17)=0;
DATA(:,18)=0.5*(DATA(:,3)+DATA(:,10));
DATA(:,19)=0.5*(DATA(:,4)+DATA(:,11));

%set the fixation indicator as follows:
%if the gaze position (x,y) over a rolling 100ms window does not change by more than
fixationhrange and fixationvrange then considered to be part of a single fixation
for efi = fixationduration:size(DATA,1)
    if sum([sum(DATA(1+efi-fixationduration:efi,9)),sum((DATA(1+efi-
fixationduration:efi,16)))]) == 0 %if all rows of tracked data have Tobii validity indicator
of 0
        if max(DATA(1+efi-fixationduration:efi,18)) -  min(DATA(1+efi-
fixationduration:efi,18))<= fixationhrange %if gaze x remains within spatial range
            if max(DATA(1+efi-fixationduration:efi,19)) -  min(DATA(1+efi-
fixationduration:efi,19))<= fixationvrange %if gaze y remains within spatial range
                DATA(1+efi-fixationduration:efi,17) = 1;
            end;
        end;
    end;
end;

%count the fixations, their average centre coordinates and duration
fixationcount = 0;
fixok = 0;
for efi = fixationduration:size(DATA,1)
    if DATA(efi,17) == 1
        if DATA(efi-1,17) == 0;
            if fixationcount > 0; %first time thru?
                FIX(fixationcount,1) = fixatedx/timepoints;
                FIX(fixationcount,2) = fixatedy/timepoints;
                FIX(fixationcount,3) = fixationlength;
                fixok = 1;
            end;
            fixatedx = 0;
            fixatedy = 0;
            fixationlength = 0;
            timepoints = 0;
            fixationcount = fixationcount + 1;
        end;
        fixatedx = fixatedx + DATA(efi,18);
        fixatedy = fixatedy + DATA(efi,19);
        fixationlength = fixationlength + 1000/60;
        timepoints = timepoints + 1;
    end;
end;
if DATA(efi-1,17) == 1; %ends on a fixation?
    FIX(fixationcount,1) = fixatedx/timepoints;
    FIX(fixationcount,2) = fixatedy/timepoints;
    FIX(fixationcount,3) = fixationlength;
    fixok = 1;
end;

%produce simple scatter plot of the fixation centres with circles sized to
%match the fixation duration at that point
if fixok == 0
    disp('NO FIXATIONS DETECTED');
else
    FIX(:,1) = 1-FIX(:,1); %flips x-coordinates to match plot axis
    FIX(:,2) = 1-FIX(:,2); %flips y-coordinates to match plot axis
    scatter (FIX(:,1),FIX(:,2),FIX(:,3));
    axis([0 1 0 1]);
end;

%write out the analysed file
csvwrite(strcat(path,'\fixations.csv'),DATA);

% ************************************************************************
%
% END
%
% ************************************************************************
```

Tobii Support contact:

SWEDEN/GLOBAL	GERMANY	NORTH AMERICA	JAPAN
+46 8 522 950 10 Phone	+49 69 2475 034-27 Phone	+1 703 738 1320 Phone	+81-3-5793-3316 Phone
support@tobii.com	support@tobii.com	support.us@tobii.com	support.jp@tobii.com
www.tobii.com	www.tobii.com	www.tobii.com	www.tobii.co.jp
Support hours: 9 am - 5 pm (Central European Time, GMT+1)	Support hours: 9 am - 5 pm (Central European Time, GMT+1)	Support hours: 8 am - 5 pm (US Eastern Standard Time, GMT-6)	Support hours: 9 am - 5.30 pm (GMT+9)

HEADQUARTERS, SWEDEN	CENTRAL EUROPE	NORTH AMERICA	JAPAN	CHINA
Tobii Technology AB	Tobii Technology GmbH	Tobii Technology, Inc.	Tobii Technology, Ltd.	Tobii Electronics Technology Suzhou Co., Ltd
Karlsrovägen 2D	Niedenau 45	510 N. Washington Street	3-4-13 Takanawa, Minato-ku	No. 678, Fengting Avenue
Box 743	D-60325 Frankfurt am Main	Suite 200 - Falls Church, VA	Tokyo 108-0074	Land Industrial Park
S-182 17 Danderyd	Germany	22046 - USA	Japan	Weiting, Suzhou
Sweden	+49 69 24 75 03 40 Phone	+1-703-738-1300 Phone	+81-3-5793-3316 Phone	Post code: 215122
+46 8 663 69 90 Phone	+49 69 24 75 03 429 Fax	+1-888-898-6244 Phone	+81-3-5793-3317 Fax	China
+46 8 30 14 00 Fax	sales.de@tobii.com	+1-703-738-1313 Fax	sales.jp@tobii.com	+86 13585980539 Phone
sales@tobii.com		sales.us@tobii.com		sales.cn@tobii.com

tobii
www.tobii.com