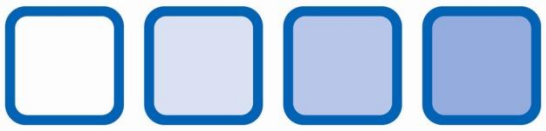
LiveTrack User Guide

Version R07 June 2016

Cambridge Research Systems Limited

80 Riverside Estate
Sir Thomas Longley Road
Rochester
Kent
ME2 4BH
England



www.crsltd.com

Part number: 071-CRS-0075_R07_LiveTrack User Guide

Version	Date	Changed by:	Description of Changes
1.0	March 2012	C. Arnold E.Wallington	First Issue
R02	September 2014	JT	Updated user guide to firmware version VET26000
R03	February 2015	JT	Updated user guide to firmware version VET26106
R04	June 2015	JT	Updated user guide to firmware version VET26110
R05	February 2016	JT	Updated to firmware version 26117 and LiveTrack Viewer version 1.0.0
R06	March 2016	JT	Updated to firmware version 26118 and LiveTrack Viewer version 1.2.0
R07	June 2016	JT	Updated to LiveTrack Viewer version 2.1.0

Introduction

The LiveTrack family of products are a range of small, self-contained eye trackers, which interface to a host computer via a USB interface. The range includes a model with an in-built camera, and models with an external video camera connected to a composite video input. An MRI compatible version is also available. LiveTrack systems can track one or both eyes (for the MRI compatible system, one camera is required per eye because of physical constraints).

The host computer does not perform the eye tracking. This is done with a high performance digital signal processor within each LiveTrack unit. The host computer therefore is just concerned with collecting the data and providing the visual stimuli.

LiveTrack uses standard USB interface classes, which makes it cross platform compatible.

Operating principle

LiveTrack illuminates the eye using one or two infrared light sources (depending on the model) to create distinct glints (reflections on the cornea) when viewed with the infrared camera. The image processing software locates the pupil as a dark elliptical region and the Purkinje glint(s) as bright elliptical regions, which lie close to the pupil.

The pupil and Purkinje centres are then determined to sub pixel precision by ellipse fitting and used to track the subject's direction of gaze while remaining tolerant to head movement and sudden changes in ambient illumination.

Calibration

To enable LiveTrack to output accurate data about the measured gaze position and pupil size a calibration is necessary. To calibrate the gaze position the subject is required to look at positions at known locations, for example dots shown on a computer screen. To calibrate the pupil size a "Scale Calibration Stick" is supplied with the LiveTrack. See further details about calibration later in this manual.

Camera video stream

LiveTrack emulates a video camera using the standard USB Video Device Class (UVC; i.e. like a webcam). This enables the camera view to be seen on the host computer, which is essential for aligning the camera with the subject. This video stream is not necessarily every video frame, but it is fast enough for alignment and focus adjustments when setting up the camera.

Data interface

LiveTrack transmits data over the USB connection along with the video stream. The data can be transmitted using either a protocol emulating a virtual serial port (CDC) or a protocol using the Human Interface Device (HID) standard that is similar to the protocol used by a keyboard or a mouse. See the corresponding chapters for details about each of these two data interfaces.

Synchronising eye tracking with other events and the video feed

LiveTrack has a trigger input which can be connected to the stimulus generating system. The state of this trigger input is recorded along with the eye tracking results so that a marker indicating the state of the trigger input accompanies every recorded sample. It is also possible to reset the frame count on the next rising edge of the trigger signal instead of placing a marker.

The video feed from LiveTrack also contains the frame-count overlaid in the upper left corner of the frame (a "burnt-in time code"). All eye-tracking results can therefore be directly related to the video stream.

It is therefore easy to reconcile the timing of the visual stimulus, the eye-tracking results, and optionally a video of the subject's eye.

Firmware updates, software and User Manuals

All software and technical support is available through the CRS web-based Support Portal (www.crsltd.com/support/login)

System Requirements

LiveTrack is compatible with most computer operating systems that support USB devices and provide drivers for UVC, HID, CDC and MSD devices.

We have tested the following operating systems (though this list is not exclusive):

Microsoft Windows: XP Service Pack 3, Windows Vista, Windows 7, 8 or 10.

Apple Mac OS X: Mavericks (10.9)*, Yosemite (10.10)* or El Capitan (10.11).
*Mac OS X 10.10 or earlier does not support USB CDC and USB UVC interfaces at the same time so it is recommended to only use the USB HID for the data interface.

Ubuntu Linux: 14.04 LTS

Hardware Installation

Consult the separate installation manual that came with your model of LiveTrack.

Software Installation

To calibrate LiveTrack, some form of visual stimulus is required. This is normally accomplished with a display device and some stimulus generation software. We recommend the use of Psychtoolbox running in MATLAB or Octave.

http://www.psychtoolbox.org

On the internal storage of the LiveTrack (see the next chapter for further instructions on how to access) you should find a calibration demo script written in MATLAB/PTB.

A stand-alone application for video viewing, calibration and data logging is available (for Windows 64-bit and Mac OS X) from the CRS Support Portal (www.crsltd.com/support/login). This application can also be used to calibrate the LiveTrack before it is used with third party software. See figures 1 and 2.

We also recommend the use of our Bits#/Display++ system to complement Psychtoolbox, as this makes triggering and stimulus calibration easier.

http://www.crsltd.com

The LiveTrack Flash Drive

The LiveTrack flash drive is used to configure LiveTrack and for firmware updates. It is of sufficient capacity for the files that are needed by LiveTrack.

How to access the flash drive

To access the LiveTrack flash drive (with the USB connected) press and hold the button for at least 5 seconds (see the installation instructions for you particular LiveTrack model for details on where this is located). LiveTrack then reboots into its bootloader, which exposes the drive as a standard "mass storage device". LiveTrack cannot eye-track when in this mode, and the other USB interfaces are disabled.

You should now find an extra disk drive connected to your computer called "LiveTrack".

The drive activity light

Near to the USB connector on LiveTrack is a drive activity light. Do not disconnect LiveTrack when this light is on. Be aware that writing to the LiveTrack drive is very slow. It is significantly slower than other storage volumes on your computer. The light will remain on for several seconds when a file is written to the disk.

Files on the flash drive

Firmware image (.ldr)*
There is a file called "VETxxxxx.ldr", where xxxxx is the version of the file. This contains the firmware image used when eye-tracking. Cambridge Research Systems periodically distribute updates to this file. Log in to our support portal to check for updates.

MATLAB calibration demo script (.zip)*
A MATLAB demo script named crsLiveTrackCalibrationDemo.m and its dependant functions are located in a zipped file. Find further details in the "Instructions.txt" file in the compressed file.

Windows INF file for CDC interface (folder)
This folder contains two files "crsltd_usb_cdc_acm.inf" and "crsltd_usb_cdc_acm.cat" which are necessary to use for setting up the CDC interface on a Windows OS. On Mac OS X and Ubuntu this step is not necessary.

Firmware configuration file (Firmware.xml)
All operating default parameters are obtained from a file named "Firmware.xml" located on the flash drive of LiveTrack. These parameters are optimally adjusted from the factory and should not be changed unless the eye tracker is used in an abnormal setting (different camera-to-eye distance than recommended for example).

LiveTrack configuration file (LiveTrack.xml)
The file "LiveTrack.xml" contains the start-up parameters for the system. You can edit the file with any text editor but make sure it is not unintentionally modified as this can make the device inoperable. The software TextEdit, which comes with Mac OS X can for example automatically change the quotation marks (") to a different type. We recommend using Xcode on Mac OS X or NotePad on Windows.

Typical contents of LiveTrack.xml are as follows:

```
<?xml version="1.0"?>
```

```
<?xml version="1.0"?>
<DEVICE_TYPE_00>
  <ETP_CONFIG>
  </ETP_CONFIG>
        <SYSTEM>
           <Entry USB_HID="ON"/>          # (ON) enable or disable USB Human Interface Device (HID) data transfer protocol
```

```
            <Entry USB_CDC="OFF"/>          # (OFF)  enable or disable USB Communications Device Class (CDC) virtual serial port
            <Entry USB_UVC="ON"/>           # (ON)      enable or disable USB Video Class (UVC) video transfer protocol
            <Entry HID_COORDINATE_SYSTEM="COORDINATE_RAW"/>
        </SYSTEM>
</DEVICE_TYPE_00>
<LIVETRACK_CONFIG>
  <TRACKING>
    # Tracked ellipse overlay on pupil and glints
    <Entry VIDEO_OVERLAY="62" />        # (62) 0 - 63, OFF=0, ON=62, UVC slider "BackLight Comp" can also set this parameter

    # Default horizontal offset of the analysed eye image (LiveTrack AP model only)
    <Entry IPD="0" />                   # UVC slider "Zoom" can also set this the parameter

    # Enable tracking left eye by default (LiveTrack FM and AP models only)
    <Entry TRACK_LEFT_EYE="ON"/>        # UVC slider "Focus" can also set this parameter

    # Enable tracking right eye by default (LiveTrack FM and AP models only)
    <Entry TRACK_RIGHT_EYE="ON"/>    # UVC slider "Focus" can also set this parameter
      </TRACKING>

# Default contents of the calibration matrices.
# Replace the following when a calibration has been performed.
  <DEFAULT_CALIBRATION>
        <LEFT_EYE>
                <Entry A11="-2.919246444803344e+15" />
                <Entry A12="7.018943162941563e+13" />
                <Entry A13="-1.598589031725037e+14" />
                <Entry A14="3.269469762188474e+14" />
                <Entry A21="2.089800231371030e+14" />
                <Entry A22="2.684058946760330e+15" />
                <Entry A23="4.379486245704038e+14" />
                <Entry A24="-9.419698422864538e+13" />
                <Entry A31="1.304294904386833e+15" />
                <Entry A32="-1.326297387626762e+15" />
                <Entry A33="1.855940866399211e+14" />
                <Entry A34="1.126825202180877e+15" />
                <Entry A41="2.609400941628260e+12" />
                <Entry A42="-2.654266828620647e+12" />
                <Entry A43="3.730264149784392e+11" />
                <Entry A44="2.251823905124515e+12" />


                <Entry RPC="1.387624860285202e+02" />
                <Entry X_SINGLE_GLINT_OFFSET="0"/>
                <Entry Y_SINGLE_GLINT_OFFSET="0"/>
        </LEFT_EYE>
        <RIGHT_EYE>
                <Entry A11="-2.919246444803344e+15" />
                <Entry A12="7.018943162941563e+13" />
                <Entry A13="-1.598589031725037e+14" />
                <Entry A14="3.269469762188474e+14" />
                <Entry A21="2.089800231371030e+14" />
                <Entry A22="2.684058946760330e+15" />
                <Entry A23="4.379486245704038e+14" />
                <Entry A24="-9.419698422864538e+13" />
                <Entry A31="1.304294904386833e+15" />
                <Entry A32="-1.326297387626762e+15" />
                <Entry A33="1.855940866399211e+14" />
                <Entry A34="1.126825202180877e+15" />
                <Entry A41="2.609400941628260e+12" />
                <Entry A42="-2.654266828620647e+12" />
                <Entry A43="3.730264149784392e+11" />
                <Entry A44="2.251823905124515e+12" />


                <Entry RPC="1.387624860285202e+02" />
                <Entry X_SINGLE_GLINT_OFFSET="0"/>
                <Entry Y_SINGLE_GLINT_OFFSET="0"/>
        </RIGHT_EYE>
  </DEFAULT_CALIBRATION>
</LIVETRACK_CONFIG>
```

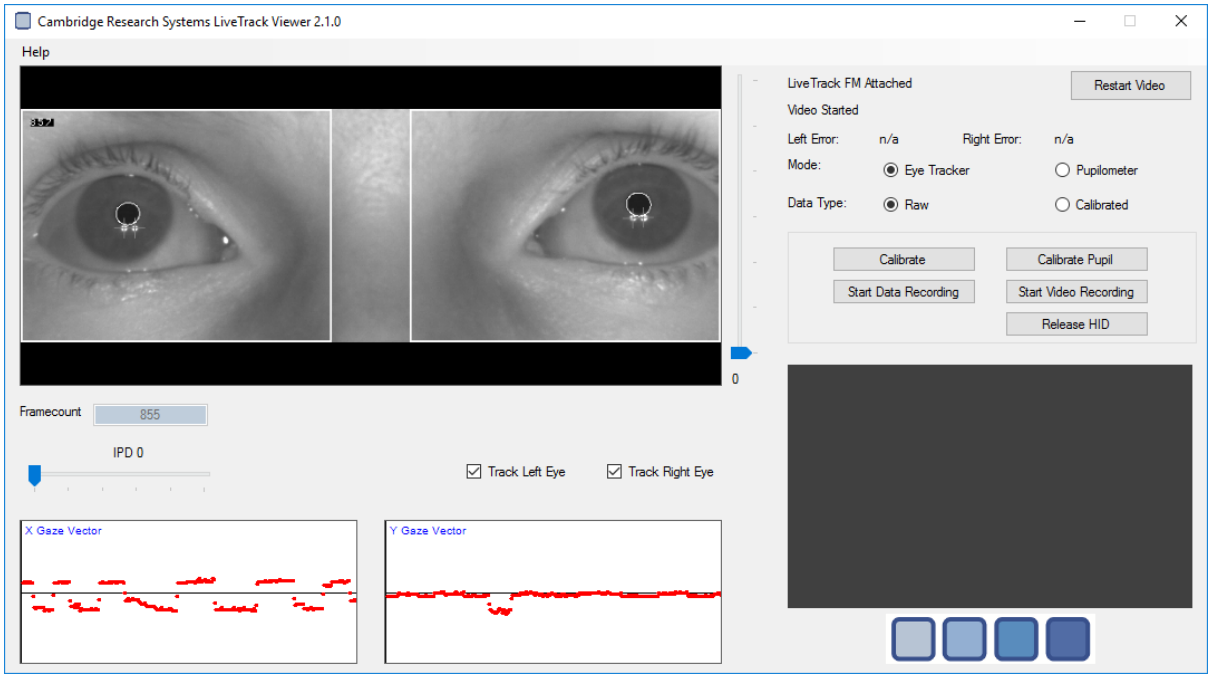


Figure 1: LiveTrack Viewer stand-alone app (Windows version) running a LiveTrack FM

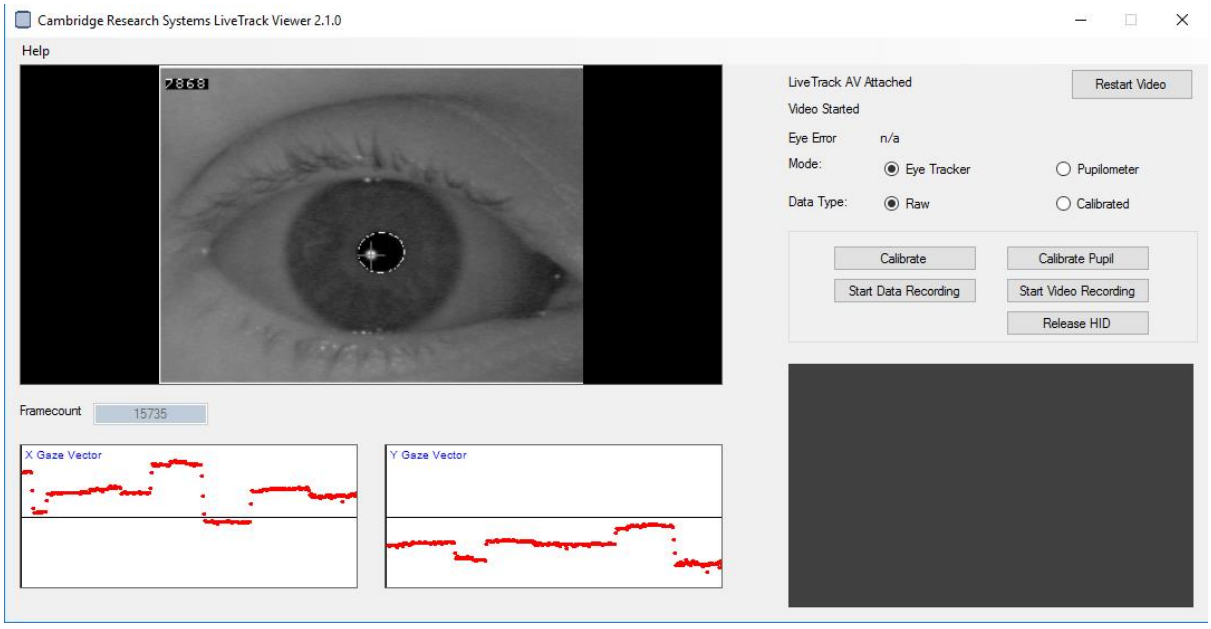


Figure 2: LiveTrack Viewer stand-alone app (Windows version) running a LiveTrack AV

Calibration Overview

Infra-Red illumination

Each LiveTrack system has one (e.g. LiveTrack AV for fMRI) or two (e.g. LiveTrack FM, LiveTrack AP) sources of infra-red illumination adjacent to the camera itself. Infra-red light is invisible to the participant so can be used to illuminate the eye without interfering with processing of visible light, often required for the current task. The infrared light is absorbed behind the pupil, therefore appearing dark to the LiveTrack system, while the corneal surface will reflect some of the infrared light. This corneal reflection is the first purkinje image (also referred to as a 'glint'), and there will be the same number of glints as there are IR illumination sources.

Limits of tracking using pupil and corneal reflections

The LiveTrack systems use algorithms to estimate the shape, size and location of the pupil and any glint(s). It then uses the relationship between these data to estimate the point of gaze. However, it should therefore be noted that tracking can only occur if these features are visible to the LiveTrack system. If gaze moves too far towards the periphery of the camera view, one or more of the features may become obscured. For example, a large vertical movement may make the pupil or glint(s) become obscured by the participant's eye-lid or lashes. Similarly, the glints are only reflected from the corneal surface for a range of central positions. An eye movement too far in any direction may cause the glints no longer to be reflected from the surface of the cornea and therefore invalid for estimating gaze position.

The setup and relationship between camera, participant and screen should therefore be taken into consideration when designing stimuli and deciding where they can be displayed appropriately.

Raw camera coordinates

The raw tracking data is measured in camera pixels (with [0, 0] being the upper left corner of the video image). Without applying some calibration data, these cannot be used to estimate direction of gaze but can still be used to detect eye movements (such as a saccade away from fixation). The raw data can be obtained with the "Start Raw Tracking" command for streaming data on the HID interface or the "$Raw" command for streaming data on the CDC ACM interface (virtual serial port).

Why we calibrate

Usually the aim of eye tracking is to obtain some information about the eye in relation to the external world, such as to estimate where a participant is looking, or to measure the magnitude of an eye movement, or pupil size. However, all of these examples require 'real world' units such as millimetres or degrees of visual angle, and probably in a different orientation to that of the camera. This will vary across different setup configurations and so LiveTrack systems cannot know these automatically. Instead, they process the image it receives in units of 'camera pixels'. These are arbitrary and of limited use in isolation. Therefore, a calibration is usually required to calculate the relationship between these camera pixel units and real world equivalents, given a particular setup.

Note that even if only a 'fixation monitor' is required, detecting any eye movements that may indicate a failure to maintain fixation, with no need to know where they actually relate to on a screen, a calibration may still be required if wanting to define the threshold acceptable limit in units of degrees of visual angle.

Also note that the calibrated spatial units used are arbitrary, but it is important to be consistent. We suggest millimetres or screen pixels, but as long as the same units are used throughout and

when calibrating, it does not matter. That is, if the calibration targets are given in millimetres the calibrated data will be in millimetres.

Calibration constants

Calibration is required to get results in real-world 3D units rather than just 2D camera pixels. The calibration procedure requires the subject to view a number of screen targets from a known distance. Cambridge Research Systems provide a MATLAB example script showing how this can be done. This allows the LiveTrack to learn the viewing geometry so that it can determine view direction for all subsequent images.

We use the term *"learn"*, deliberately. The derivation of the calibration matrices is an iterative process that can be sensitive to initial conditions.

Once this is done a calibration matrix (one for each eye in binocular LiveTrack versions) is generated. The gaze location of the screen or the direction of gaze can then be returned in real world distances or angular units (Helmholtz or Fick coordinates).

Note: LiveTrack always returns degrees for angle units (Helmholtz or Fick coordinates).

$$\begin{bmatrix} A11 & A12 & A13 & A14 \\ A21 & A22 & A23 & A24 \\ A31 & A32 & A33 & A34 \\ A41 & A42 & A43 & A44 \end{bmatrix}$$

A-Matrix

This matrix is used in the overall model converting raw position data to calculation of gaze position and does not necessarily relate to a specific unit. The A-matrix is applied to the apparent gaze vector (pupil position minus centre of cornea curvature) and does rotation, perspective correction and scaling of the apparent gaze vector.

How to Calibrate

In the following discussion the pupil position is given by the coordinates (pX, pY, pZ), and the centre of corneal curvature (gX, gY, gZ). These are both in units of camera-pixels. (x, y, z) is the target position on the screen and is appropriate real world units (e.g. mm).

Note that if angular units are to be used after calibration you must make the subject's eye the origin [0,0,0].

The procedure for calibration should be as follows:

1. **Decide what constitutes a fixation**

   Before presenting calibration targets, it needs to be decided what parameters will be used to constitute a fixation. The eyes are always moving – even when fixating, they are not truly stationary but making micro saccades around the target location. Therefore, it needs to be decided what the maximum distance (in camera pixels) between recorded eye position data is that will be tolerated before it is concluded that fixation has been achieved or maintained. If the maximum distance between eye position data in a given sample duration is less than this threshold limit then it will be concluded that the participant is fixating on that target.

   The duration of the sample also needs to be decided in advance. It is possible that the participant may be making a large eye movement that passes through the target location briefly but continues on to its eventual location elsewhere. This would not constitute a

fixation and so the minimum duration that consecutive eye position data must remain within the error window specified above needs to be set.

If either of these parameters are too relaxed (such as a large error window or short minimum duration), then a fixation may be erroneously concluded when eye movements while participants were still making relatively large eye movements. However similarly if these parameters are set to be too stringent (such as a very narrow fixation window or a very long fixation time), then calibration may take some time as participants may have difficulty meeting these standards.

Unfortunately, at this stage, it cannot be known how many visual degrees each camera pixel relates to, so the decision of the size of the error window is largely arbitrary and may require some pilot data to make an informed decision.

2. **Present targets at known locations**

   In order to calibrate camera pixels with 'real world' coordinates, eye position data needs to be recorded while the participant is fixating on a target at a known location. Typically, 9 targets are used, presented in a 3x3 grid, however any number of dots could be used depending on the user's requirements.

   It is important to note that although any system of coordinates can be used (such as distance from centre of screen, distance from one corner of the screen, distance from the centre of the fixation matrix, etc.), the origin of this system will be the origin of all subsequent calibrated results. If choosing to make an arbitrary origin (such as the centre of the calibration matrix which is presented at a location below the centre of the screen due to head position and trackable visual field), then this offset should be applied to all subsequent calibrated results too.

3. **Calculate an error term for all the points**

   These values can then be entered into the equation shown in below, which give the total error between predicted and actual locations of each target fixation:

$$aXYZW = \begin{bmatrix} A_{11} & A_{12} & A_{13} & A_{14} \\ A_{21} & A_{22} & A_{23} & A_{24} \\ A_{31} & A_{32} & A_{33} & A_{34} \\ A_{41} & A_{42} & A_{43} & A_{44} \end{bmatrix} * \begin{bmatrix} pX\text{-}gX \\ pY\text{-}gY \\ \left(1\text{-}\sqrt{(pX\text{-}gX)^2 + (pY\text{-}gY)^2}\right) \\ 1 \end{bmatrix}$$

$$oXYZW = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$Err = \sum \left(\frac{aXYZW(1:3)}{aXYZW(4)} \text{-} oXYZW \right)^2$$

Note, that first number in the term "1-sqrt((pX-gX^2+(pY-gY)^2)", which here is shown as one (1) is actually the distance between the centre of the corneal curvature and the centre of the pupil , measured in camera pixels. The actual value is typically in the range of 50 – 200. It is not critical to use an acurate value for this parameter as long as the same value is used with the the same calibration matrix. It can however be a good idea to set it to around 100 (or try to estimate it more acurately if the minimisation fails to compute the appropriate calibration matirx (see below).

4. **Use a numerical minimisation technique to find the A-matrix**

   The aim is to find the matrix parameters that result in the minimal total error (i.e. the sum of the squared error for each target). There are many methods and functions designed to do this. The method used in the MATLAB calibration demo is to use the function fminsearch.

5. **Check the error term for each calibration point individually**

   Once the optimal matrix values have been estimated (to give the minimal error), these matrices can be used to convert all subsequent measurements in camera-pixel units into real-world units. However, although the error may be minimised with these parameters, this is only for the target and eye position data given to it. Before this matrix is used in actual trials, it should first be checked how large this error is for each target. For example, if a participant failed to fixate on each target properly, instead fixating at some other location, then the matrix will give incorrect predictions. Although the matrix may have values which give the minimal error, this error may still be too great to give meaningful results and the calibration process may need to be repeated.

What to do with the calibration parameters obtained

Both the CDC and HID interfaces allow the calibration parameters to be uploaded back to LiveTrack (see Appendix A and B for details). These values then take effect immediately, but are not permanently stored.

Alternatively, the default parameters in the xml file can be updated, which will take effect when LiveTrack is restarted.

Either method then allows angular or spatial results to be returned by either the CDC or HID interfaces.

The calibration parameters can also be applied directly to the raw data on the host computer to obtain the calibrated gaze vector:

$$oXYZW = \frac{aXYZW(1:3)}{aXYZW(4)}$$

Where aXYZW is the same as before and oXYZW is the calibrated gaze vector.

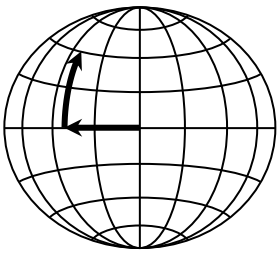
Spatial results

LiveTrack returns [x,y,z] coordinates of a point in the same plane as used for the calibration procedure. The scaling (units) and origin will be that used during calibration.

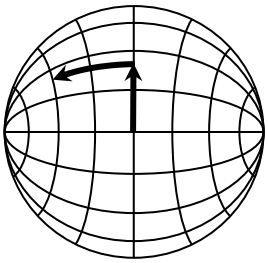
Helmholtz or Fick angular coordinates

LiveTrack returns the latitude and longitude or azimuth and elevation of the gaze direction in degrees. LiveTrack internally calculates the spatial results as a precursor to the angular units. Therefore, the origin and orientation of the spatial units used when calibrating determine those of the angular units. Usually only makes sense to make the eye the origin if you are returning angular units.

The Fick system of angular coordinates specifies the horizontal (longitude) component first followed by the vertical (latitude) component. The Helmholtz system specifies the vertical (elevation) component first followed by the horizontal (azimuth) component. These coordinate systems are demonstrated in the figure below:



Fick Coordinates



Helmholtz Coordinates

It is important to be aware of which angular coordinate system is being used, as their angular components do not commute! For example, a longitude of 50 degrees followed by a latitude of 30 degrees is not equivalent to an elevation of 30 degrees followed by an azimuth of 50 degrees.

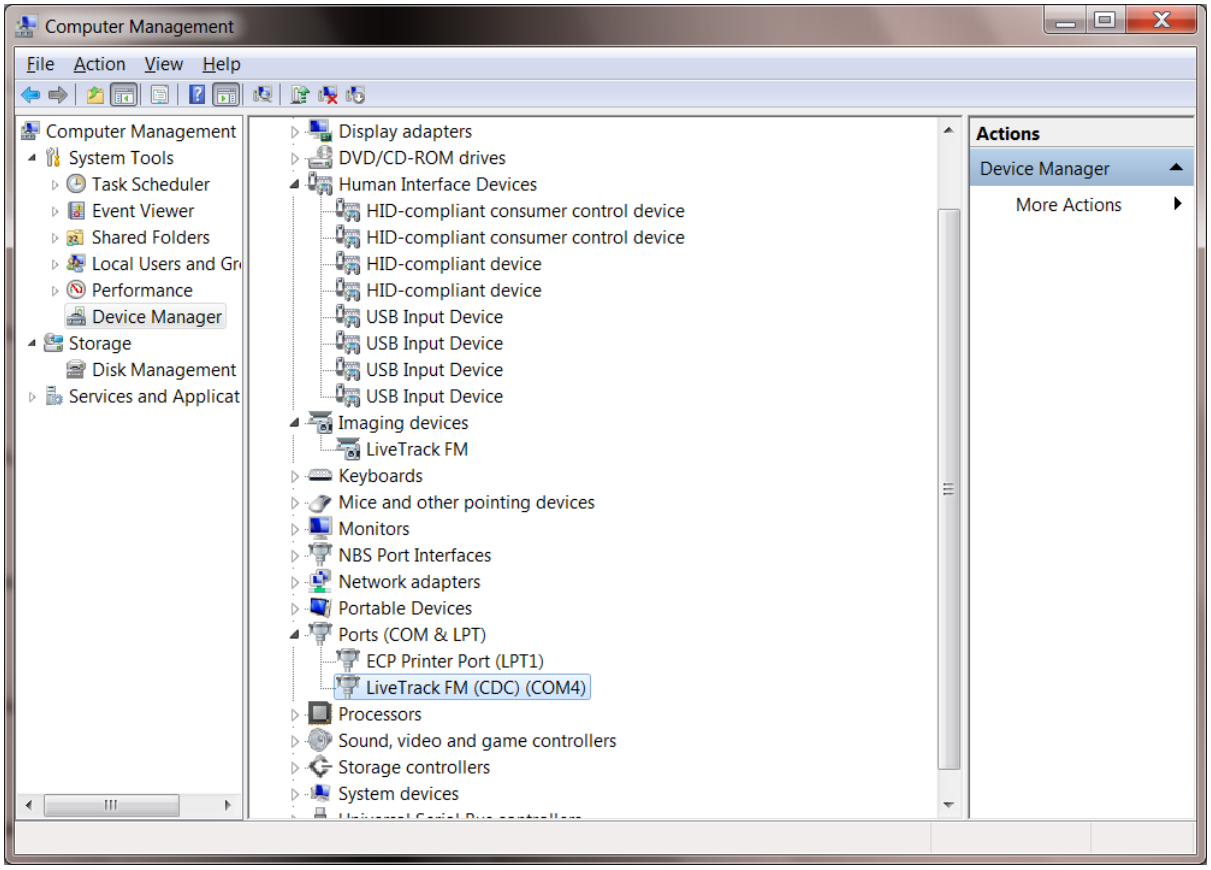
Using the serial "CDC" interface with Windows 7 or newer

The LiveTrack system is connected to the computer via a USB port but communication occurs as though it were a serial port. It is possible to configure the USB connection to behave as a serial-like port. For this step it is necessary to use the INF file located on the Mass Storage Device (MSD) drive.

Setting up the COM port

1. Locate the folder "Windows INF file for CDC interface" on the MSD and copy it to the local disk drive
2. Open LiveTrack.xml in NotePad and change setting for USB_CDC to "ON"
3. Eject the MSD drive
4. The device should now appear with the CDC ACM interface and Windows is likely to prompt you with the Windows Add New Hardware wizard. Follow the instructions and choose "Browse" to direct the wizard the INF file that you stored in step 1. After installation you should be able to see which COM port number the device has been assigned
5. If you are not able to see the COM port go the device manager to see the COM port number there:
    a. Click Start
    b. Right-click on 'My Computer' and select 'manage'
    c. Select 'Device Manager' from the list on the far left.
    d. Locate 'Ports' and look for "LiveTrack FM" or "LiveTrack AV"

If the wizard did not appear check in Device Manager that it is installed correctly. If not, right-click on the device and click "Update Driver Software" and guide the wizard to the INF setup file.


Computer Management
File Action View Help
Computer Management
System Tools
Task Scheduler
Event Viewer
Shared Folders
Local Users and Gr
Performance
Device Manager
Storage
Disk Management
Services and Applicat
Display adapters
DVD/CD-ROM drives
Human Interface Devices
HID-compliant consumer control device
HID-compliant consumer control device
HID-compliant device
HID-compliant device
USB Input Device
USB Input Device
USB Input Device
USB Input Device
Imaging devices
LiveTrack FM
Keyboards
Mice and other pointing devices
Monitors
NBS Port Interfaces
Network adapters
Portable Devices
Ports (COM & LPT)
ECP Printer Port (LPT1)
LiveTrack FM (CDC) (COM4)
Processors
Sound, video and game controllers
Storage controllers
System devices
Actions
Device Manager
More Actions

Using PuTTY

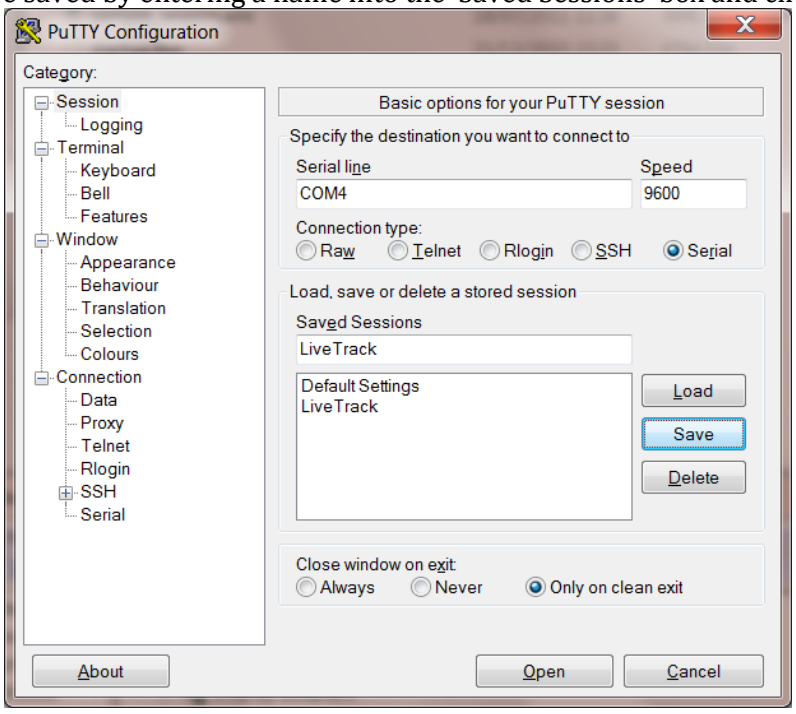
Setting up a connection

1. Windows 7, 8 and 10 do not include a 'hyperterminal' as XP does, however PuTTY is a free, open source program that can be used to interface with The LiveTrack system. It can be downloaded from www.putty.org.

2. When installed and opened, change the 'connection type' to 'Serial' (see image below).

3. After changing the connection type to serial, enter the name of the port The LiveTrack system is connected to (COM4 in this example). If you are unsure, check in Device Manager (see above).

4. The speed in bytes per second can be specified, but unless reason to change it, as the serial port is simulated, it can be left as the default value. For future convenience, these setting can be saved by entering a name into the 'Saved Sessions' box and clicking 'save'.



5. The LiveTrack system should now be connected via a serial-like port and the main PuTTY window should be open and ready to be written to.

6.

Communicating with The LiveTrack system

Once the serial port has been created, commands can be types into PuTTY. Commands are entered as words, always preceded a dollar ($). To see a full list of possible commands, either see the appendix B of this manual or type:

```
>> $Help
```

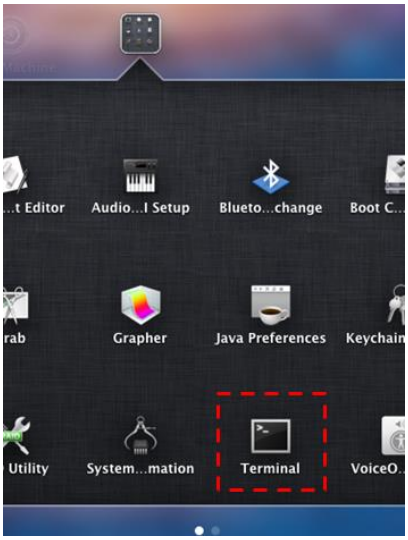
Using the serial "CDC" interface with OSX.

The LiveTrack system is connected to the computer via a USB port but communication occurs as though it were a serial port. It is possible to configure the USB connection to behave as a serial-like port. Note that OS X 10.10 and earlier does not support CDC at the same time as UVC (Video) so it is recommended to use the HID data interface instead of CDC if not using Mac OS X 10.11 (El Capitan).

Setting up a connection

1. Open LiveTrack.xml located in the LiveTrack MSD drive in NotePad
2. Change setting for USB_CDC to "ON"
3. Eject the MSD drive
4. The device should now appear with the CDC ACM interface and ready to communicate with
5. The LiveTrack system can be communicated with using the terminal. To use this, open 'Launchpad' in the bottom left of the screen.

1. Select Utilities and then click on 'Terminal' (see images below).



2. Regardless of which directory Terminal may be in when you open it, to get to the directory needed, type:

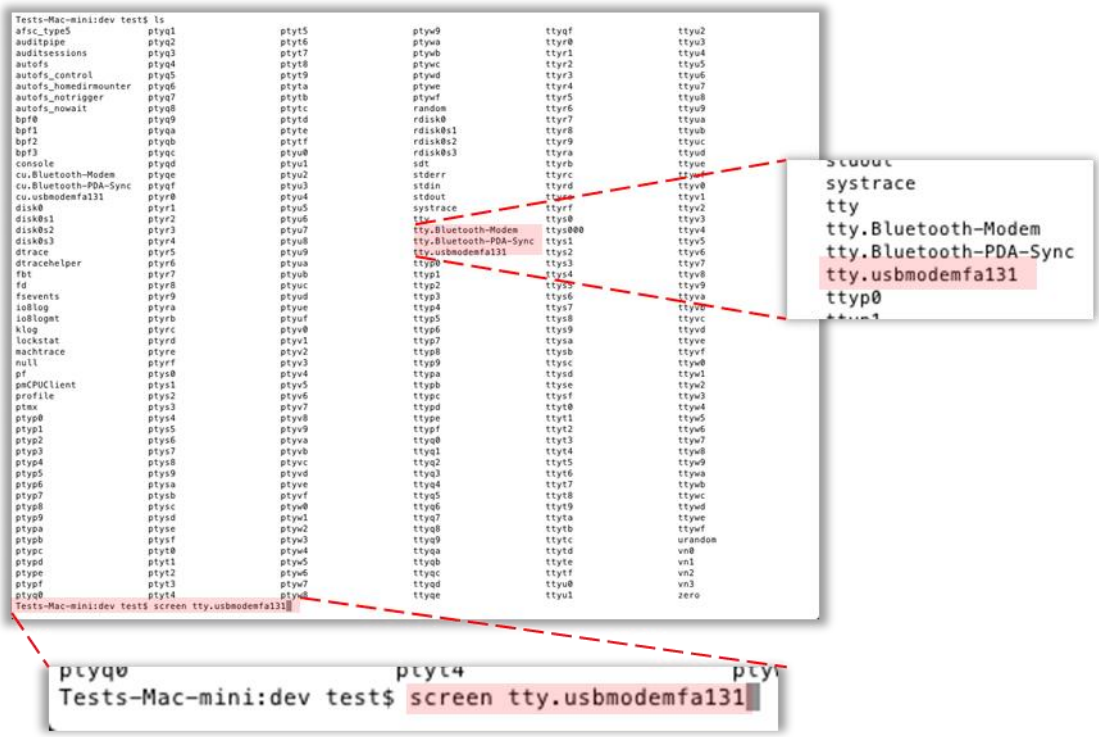
```
>>cd /dev
```

3. Type:

   ```
   >>ls
   ```

   and from the list that it presents, find the name of the location The LiveTrack system is connected to ('tty.usbmodemfa131' in the below example). The name will always start tty.usbmodem. It is then followed by a number
4. Finally, type 'screen' followed by the name of the port LiveTrack is connected to. For example:

   ```
   >> screen tty.usbmodemfa131
   ```



5. This should then blank the previous input commands and any output they displayed, leaving a blank screen. This is the open connection with the LiveTrack system.

Communicating with the LiveTrack system

Note that you will not be able to see what you type as you type it, but it will register your key presses. If you enter a command and nothing happens, try typing it again taking care to ensure correct spelling and character case.

Once the serial port has been created, commands can be typed into the terminal. Commands are entered as words, always preceded by a dollar ($) symbol. To see a full list of possible commands, either see the appendix of this manual or type:

```
>> $Help
```

Closing the connection with The LiveTrack system

To exit the 'Screen' function (freeing up the serial port for future use in other scripts), press 'Ctrl' + 'A', followed by 'Ctrl' + '\'. A black box should appear at the bottom of the screen asking if you are sure you want to quit. Press 'Y' if yes.

To exit the terminal and return to the desktop, either use a 3-finger swipe across a track pad from left to right. This will leave terminal open in the background and can be returned to by a 3-finger swipe in the opposite direction. Alternatively, to close terminal completely, press 'Cmd' + 'Q' (The Cmd key is the one with the symbol ⌘).

Using the serial "CDC" interface and MATLAB

Note: it is recommended to use the HID data interface in MATLAB with the free toolbox "Psychtoolbox" and its function "PsychHID".

Introduction

In MATLAB, information is passed to, and read from, the LiveTrack system the same way as though writing to, or reading from, a file using *fopen, fprintf, fscanf* and *fclose.* Commands and information are passed in the form of strings rather than numbers. Although most of this can be done using the hyperterminal/terminal/PuTTY, some uses may need to communicate with the LiveTrack system in the middle of a script.

Creating a serial port object and handle

First, create a serial port object and handle where *port* is the name of the USB port the LiveTrack system is connected to. If you are unsure of this, see the 'setting up the LiveTrack system' chapter above, appropriate for your operating system.

```
>> s1 = serial(' port ');
```

The syntax of serial ports varies across platforms. Some examples include:

```
>> s1 = serial('com1');                    % MicrosoftWindows 32 and 64-bit
>> s1 = serial('/dev/tty.usbmodemfa121');  % Mac OS X and Mac OS X 64-bit
>> s1 = serial('/dev/ttyS0');              % Linux 32 and 64-bit
```

Opening the serial port for communication

Once it is created, to open the port for communication, use:

```
>> fopen(s1);
```

Checking the current status of the serial port

The current status of the serial port can be checked at any time by simply typing the name of the serial port object (e.g. s1) without a semicolon.

Writing to the LiveTrack system via the serial port

While open, the LiveTrack system can be written to using *fprintf.* For example, to start recording eye tracking data:

```
>> fprintf(s1, ['$Raw' 13]);
```

All The LiveTrack system commands start with a dollar ($) symbol. Also, the '13' is the terminator character (13 is a carriage return) and should be added to the end of all input commands.

Reading from The LiveTrack system via the serial port

Some input commands will generate a reply from The LiveTrack system which can be read using *fscanf*. One can check whether there is information available to be read using:

```
>> s1.BytesAvailable
      ans = 0
```

Positive values indicate available information. For example, '$Help' should return a list of possible commands, including the different modes The LiveTrack system can be set to (see below):

```
>> fprintf(s1, ['$Help' 13]);
>> s1.BytesAvailable
```

```
      ans = 117
>> helpOutput = fscanf(s1);
>> helpOutput
```

This may only return one line (one command) at a time. To read out the whole buffer use:

```
>> helpOutput = fread(s1, s1.BytesAvailable);
```

Both the input and output buffers have a limited size (default = 512 bytes). Attempting to write or read more bytes than there is space will result in an error. The current size of the buffers can be checked using either:

```
>> s1.InputBufferSize
        ans = 512
>> s1.OutputBufferSize
        ans = 512
```

If larger buffers are needed, these sizes can be changed. For example:

```
>> s1.InputBufferSize = 1000;
>> s1.OutputBufferSize = 1500;
```

Closing the Serial Port

Once the serial port has finished being written to, it should be closed as soon as possible to minimise the chances of errors occurring if the script aborts part way with the port still open:

```
>> fclose(s1);
```

If a script does abort without closing the port properly, the port can get stuck open. It may have no handle assigned to it with which to close it, and further attempts to reopen it will likely fail as it is already open. Closing and then opening MATLAB again should clear all ports though, allowing for it to then be used again.

Disconnect the Serial Port

Once a port is finished with (such as at the end of a script), it should be disconnected and closed properly to avoid potential problems when trying to open it again in future:

```
>> delete(s1);
>> clear s1;
```

Using the "HID" interface

Overview

Data is sent to or from the LiveTrack system in the form of 'reports'. Each report is an array of 64-bytes (therefore values between 0 and 255) and each byte *or a group of bytes* represents the value of a parameter (e.g. pupil diameter). If tracking is enabled in the XML file, then these

reports are sent continuously as soon as the LiveTrack system is connected. This can be stopped by passing it the 'stop tracking' command and started again by passing the 'start tracking' command. These initial reports saved within the buffer may need to be cleared when starting a trial so that only reports relevant to the users' current behaviour are read and saved.

On LiveTrack FM versions (binocular tracking enabled) the left and right eye tracking data will be returned in one single HID report. However, on LiveTrack AV versions which are optimized for monocular tracking on video feeds from PAL or NTSC cameras, the "left" and "right" eye data are replaced with tracking from two consecutive samples from the same eye. Note that binocular tracking can still be achieved on LiveTrack AV by using two cameras and two LiveTrack DSP modules.

Combining multiple bytes

Most parameters are represented by 2 bytes which combine to give a single overall value. The bytes for each parameter are in ascending order, from least significant (LS) to most significant (MS).

To combine these to give the actual value for that parameter, multiply the value in the right (MS) byte by 256 and then add the value in the left (LS) byte. So for example:

[101 12] = (12*256) + 101 = 3173

However some exceptions exist. For example, the timestamp value works in a similar way but with 8 bytes. The same equation as with 2 bytes can be used, but the value in each successive byte needs to be multiplied by an additional 256. For example:

1st byte + (2nd byte * 256) + (3rd byte * 256 * 256) + (4th byte * 256 * 256 * 256)…

Using the HID interface with Psychtoolbox

MATLAB users can conveniently use the function PsychHID included in the open source toolbox Psychtoolbox (www.psychtoolbox.org).

Extra steps required if using Windows

PsychHID does not automatically work if using Microsoft Windows and attempts to use it will return an error message stating that a .mexw32 file could not be found, even though the file does exist in the directory it specifies. The problem is another file, libusb-1.0.dll.

Run LoadPsychHID. If you get an error message follow the instruction for installation.

Establishing LiveTrack's device number

Before reading or writing data from the LiveTrack, its deviceNumber needs to be established. PsychHID assigns an index to all HID enabled devices attached to the computer. To see this list, and therefore search it for the LiveTrack index, use:

>> deviceList = PsychHID('Devices');

This will return a structure with each HID device assigned its own position. To identify the LiveTrack system, cycle through the structure and check the 'product' field. This will say 'LiveTrack FM' , 'LiveTrack AP' or 'LiveTrack AV' (depending on the specific product used) if it is the LiveTrack system.

Once the deviceNumber has been established, the device can be communicated with using this value as the deviceNumber.

Setting Reports

To send a command to LiveTrack, such as to start or stop tracking, use:

>> err = PsychHID('SetReport', *deviceNumber, reportType, reportID, report*);

reportType can be a value between 0 and 3. For the current purposes, this should be set to 2, which represents output (from the host computer, not from the Livetrack system), as data is being written to the LiveTrack system

reportID should be set to 0.

report is the report bytes. Even if only a few bytes are used always send a complete 64 byte report. Unused bytes can be set to 0. The first two bytes correspond to the command type, with the least significant byte on the left and the most significant on the right. Therefore the first byte should be set to value that corresponds with the command to be sent (see table below) and the right hand byte should be set to 0. Bytes 3 and 4 represent the 'tag' value. These can also be set to 0 by default although may be used by some commands. Note that the values sent should be converted to 8 bit integer in MATLAB with the uint8.

Reports sent to LiveTrack

Command	reportID value	Explanation (see appendix for detail)
Properties	100	Change the configuration of LiveTrack.
Stop Tracking	102	Will stop sending reports to the host computer.
Reset Timestamp	103	Will reset the frame counter (timestamp).

Start Raw Tracking	104	LiveTrack starts tracking and returns results in camera pixel units.
Start Calibrated Tracking	105	LiveTrack starts tracking and returns results in calibrated units.

Therefore to start tracking (while leaving the tag value as 0), use:

>> err = PsychHID('SetReport', *deviceNumber,* 2, 0, uint8([104 zeros(1,63)]));

And to stop tracking (while leaving the tag value as 0), use:

>> err = PsychHID('SetReport', *deviceNumber*, 2, 0, uint8([102 zeros(1,63)]));

Reading Reports

To get report data using PsychToolbox, use:

>> [report, err] = PsychHID('GetReport', *deviceNumber*, *reportType*, *reportID*, *reportBytes*);

reportType can be a value between 0 and 3. For the current purposes, this should be set to 1, which represents input (to the host computer, not to the livetrack system) as data is being read.

reportID can be an integer between 0 and 255. This value is largely arbitrary but can be used to identify and differentiate between the reports later.

reportBytes is the number of bytes to read before stopping. As the report will have 64-bytes, this should be the maximum value given for this argument.

Therefore, to read the entire report from a LiveTrack system assigned deviceNumber 2 and to assign a reportID of 100 to the report:

>> [report, err] = PsychHID('GetReport', 2, 1, 100, 64);

The returned 'report' variable will be the report array. The 'err' variable is the returned error. This should be 0 if the report was read successfully. If this is not 0, check the command has been typed correctly.

Reports received from LiveTrack

Command	reportID value	Explanation (see appendix for detail)
Raw tracking results	200	Results in camera pixel units.
Calibrated tracking results	201	Results in calibrated units.

Interpreting the Report

If the LiveTrack is tracking, it will be sending tracking reports to the host computer, which can be read.

After a report is read from the LiveTrack system, it could then be written to a text file for later analysis. However, often paradigms may be at least partly gaze-contingent, doing different things depending on the participant's current gaze position. For example, it may be intended that a trial should not start until a participant is fixating. Therefore, it may be important to analyse the data in real time.

The 'Flag' parameter

The LeftFlag (bytes 17) and RightFlag (bytes 41) bytes are different to the other bytes in that it is not the value itself that is important, but the status of the first 6 **bits** that this value translates

to. For example, a value of 31 represents ‘011111’ and the value 19 represent ‘010011’. The ‘first’ bit (Bit 0) is the one on the far right and the index (2nd, 3rd, etc.) ascends towards the left.

If wanting to check the value of each individual bit, there are various binary calculators available that will convert a decimal value to its binary equivalent. For example, in Matlab:

```
>> dec2bin(25)

        ans = '011001'
```

However for most purposes, I you simply want to check if the eye has been tracked or not it is sufficient to check if the value is 23 (one glint system; LiveTrack AV) or 47 (two glint system; LiveTrack FM). See more details about the flag parameter in the appendix.

Appendix A: HID Reports

HID devices are natively supported by all current operating systems. Keyboards mice and joysticks are usually USB HID devices. As an eye tracker is by definition a true “human interface”, it fits quite nicely with the protocol.

Report ID numbers:

The LiveTrack HID protocol consists of one report which is read-only (Get Report) and four reports which are write-only (Set Reports).

Get Reports:
PT_RAW_RESULTS_DATA = 200

PT_CALIBRATED_RESULTS_DATA = 201

Set Reports:
PT_TRACKING = 100
PT_STOP_TRACKING = 102
PT_ARM_TIMESTAMP = 103
PT_RAW_TRACKING = 104
PT_CALIBRATED_TRACKING = 105

List of HID Reports

Note that there is a "tag" value associated with some of the reports. This is used to differentiate between multi-packet reports.

PT_TRACKING (configure the eye tracker)

Bytes	Total Bytes	Parameter	Value
1-2	2	PacketType	100
3-4	2	Tag	0
5	1	TrackLeftEye	Enable tracking left eye
6	1	TrackRightEye	Enable tracking right eye
7-8	2	ReturnedDataFormat	1 for Raw data (PT_RAW_RESULTS_DATA), 2 for Calibrated data (PT_CALIBRATED_RESULTS_DATA)
9-10	2	nFindGlints	Number of glints to find
11-12	2	DebugImage	Controls the presentation of debug images
13-14	2	Overlay	Controls the overlaid images
15-18	4	CamPixToMM	Camera pixels to world mm * 1024 (used for scaling of pupil area)
19-64	46	Unused	

PT_STOP_TRACKING (stops tracking)

Bytes	Total Bytes	Parameter	Value
1-2	2	PT_STOP_TRACKING	102, 0
3-64	62	Unused	0

PT_TIMESTAMP (reset timestamp)

Bytes	Total Bytes	Parameter	Value
1-2	2	PT_TIMESTAMP	103, 0
3-64	62	Unused	0

PT_RAW_TRACKING (starts raw pupil and glint tracking data)

Bytes	Total Bytes	Parameter	Value
1-2	2	PT_RAW_TRACKING	104, 0
3-64	62	Unused	0

PT_CALIBRATED_TRACKING (starts calibrated tracking data)

Bytes	Total Bytes	Parameter	Value
1-2	2	PT_CALIBRATED_TRACKING	105, 0
3-64	62	Unused	0

PT_RAW_RESULT_DATA (this is all packed into one packet)

Bytes	Total Bytes	Parameter	Explanation
1-2	2	PacketType	200,0
3-4	2	Tag	0
5-6	2	Digital_IO	The trigger input
7-8	2	unused	
9-16	8	FrameNumber	Frame count
17	1	LeftFlag	See explanation below
18	1	unused	
19-20	2	LeftVideoOffsetX	Offset of the ROI UVC stream compared to full image
21-22	2	LeftVideoOffsetY	Offset of the ROI UVC stream compared to full image
23-24	2	LeftCameraScaling	The 'units' of the following values are scaled by the factor indicated here. For actual units, the following values should be divided by this value
25-26	2	LeftPupilWidth	The width of the left pupil (will be 0 if not tracked this frame)
27-28	2	LeftPupilHeight	The height of the left pupil (will be 0 if not tracked this frame)
29-30	2	LeftPupilCameraX	The (camera) X-coordinate of the left pupil
31-32	2	LeftPupilCameraY	The (camera) Y-coordinate of the left pupil
33-34	2	LeftGlint1CameraX	The (camera) X-coordinate of glint 1 of the left eye
35-36	2	LeftGlint1CameraY	The (camera) Y-coordinate of glint 1 of the left eye
37-38	2	LeftGlint2CameraX	The (camera) X-coordinate of glint 2 of the left eye
39-40	2	LeftGlint2CameraY	The (camera) Y-coordinate of glint 2 of the left eye
41	1	RightFlag	See explanation below
42	1	unused	
43-44	2	VideoOffsetX	Offset of the ROI UVC stream compared to full image
45-46	2	VideoOffsetY	Offset of the ROI UVC stream compared to full image
47-48	2	RightCameraScaling	This Is the same as LeftCameraScaling (bytes 23-24) but for the right eye
49-50	2	RightPupilWidth	The width of the right pupil (will be 0 if not tracked this frame)
51-52	2	RightPupilHeight	The height of the right pupil (will be 0 if not tracked this frame)
53-54	2	RightPupilCameraX	The (camera) X-coordinate of the right pupil

55-56	2	RightPupilCameraY	The (camera) Y-coordinate of the right pupil
57-58	2	RightGlint1CameraX	The (camera) X-coordinate of glint 1 of the right eye
59-60	2	RightGlint1CameraY	The (camera) Y-coordinate of glint 1 of the right eye
61-62	2	RightGlint2CameraX	The (camera) X-coordinate of glint 2 of the right eye
63-64	2	RightGlint2CameraY	The (camera) Y-coordinate of glint 2 of the right eye

Flag code (bytes 17 and 41 in PT_CALIBRATED_RESULT_DATA and PT_RAW_RESULT_DATA)

Bit	Name	Value = 0	Value = 1	Explanation
0	Present	Disabled	Enabled	It is possible to use the LiveTrack system to monitor either one eye at a time or both eyes simultaneously. The bit represents whether monitoring for its respective eye is enabled or disabled
1	Found Pupil	no	yes	This indicates whether the pupil was successfully tracked for that frame or not. On some frames it is possible that the eye tracking is lost, such as if the eye moves too far to one extreme that the pupil or glints are no longer visible to the camera. If tracking is lost, any data returned for this frame is unreliable and should not be used in the main analyses.
2	Found Glint 1	no	yes	At least one glint was found
3	Found Glint 2	no	yes	Glint number 2 was found (if applicable)
4	Searching for 1 glint	no	yes	How many glints LiveTrack is looking for ,encoded in two bits: '01' = 1 glint and '10' = 2 glints
5	Searching for 2 glints	no	yes	
6-7	Unused			

PT_CALIBRATED_RESULT_DATA (this is all packed into one packet, calibrated eye position data)

Note: The scaling factor is always 32 for calibrated results.

Bytes	Total Bytes	Parameter	Explanation
1-2	2	PacketType	201,0
3-4	2	Tag	0
5-6	2	DigitalIOx	The trigger input.
7-8	2	unused	
9-16	8	Frame number	Frame count
17-18	2	LeftFlag	(see description after raw results report)
19-20	2	VideoOffsetX	Left eye ROI stream X offset compare to full eye image
21-22	2	VideoOffsetY	Left eye ROI stream Y offset compare to full eye image
23-24	2	PupilWidth	The width of the left pupil (will be 0 if not tracked this frame).
25-26	2	PupilHeight	The height of the left pupil (will be 0 if not tracked this frame).
27-28	2	GazeX	Left gaze projected onto viewing screen X dimension
29-30	2	GazeY	Left gaze projected onto viewing screen Y dimension
31-32	2	GazeZ	Left gaze projected onto viewing screen Z dimension
33-34	2	GazeAzimuth	Left helmholtz azimuth coordinate of gaze
35-36	2	GazeElevation	Left helmholtz elevation coordinate of gaze
37-38	2	GazeLongitude	Left fick longitude of gaze
39-40	2	GazeLatitude	Left fick latitude of gaze
41-42	2	RightFlag	(see description after raw results report)
43-44	2	VideoOffsetX	Right eye ROI stream X offset compare to full eye image
45-46	2	VideoOffsetY	Right eye ROI stream Y offset compare to full eye image
47-48	2	PupilWidth	The width of the right pupil (will be 0 if not tracked this frame).
49-50	2	PupilHeight	The height of the right pupil (will be 0 if not tracked this frame).
51-52	2	GazeX	Right gaze projected onto viewing screen X dimension
53-54	2	GazeY	Right gaze projected onto viewing screen Y dimension
55-56	2	GazeZ	Right gaze projected onto viewing screen Z dimension
57-58	2	GazeAzimuth	Right helmholtz azimuth coordinate of gaze

59-60	2	GazeElevation	Right helmholtz elevation coordinate of gaze
61-62	2	GazeLongitude	Right fick longitude of gaze
63-64	2	GazeLatitude	Right fick latitude of gaze

PT_CALBRATION_PARAMETERS (this is actually 6 packets, three for each eye, "tag" is 1, 2, 3, 4, 5 or 6). Note that the parameters E11 through E44 are not used and can be set to 0 (but must the set). The parameters X_singleGlintOffset and Y_singleGlintOffset should always be set to zero (0).

The following three packets are sent for each eye:

Bytes	Total Bytes	Parameter	Explanation
1-2	2	PacketType	45,1
3-4	2	Tag	1 (left eye) or 4 (right eye)
5-8	4	Scale	Scaling factor
9-12	4	A11	A matrix element
13-16	4	A12	A matrix element
17-20	4	A13	A matrix element
21-24	4	A14	A matrix element
25-28	4	A21	A matrix element
29-32	4	A22	A matrix element
33-36	4	A23	A matrix element
37-40	4	A24	A matrix element
41-44	4	A31	A matrix element
45-48	4	A32	A matrix element
49-52	4	A33	A matrix element
53-56	4	A34	A matrix element
57-60	4	A41	A matrix element
61-64	4	A42	A matrix element

Bytes	Total Bytes	Parameter	Explanation
1-2	2	PacketType	45,1
3-4	2	Tag	2 (left eye) or 5 (right eye)
5-8	4	A43	A matrix element
9-12	4	A44	A matrix element
13-16	4	E11	E matrix element
17-20	4	E12	E matrix element
21-24	4	E13	E matrix element
25-28	4	E14	E matrix element
29-32	4	E21	E matrix element
33-36	4	E22	E matrix element
37-40	4	E23	E matrix element
41-44	4	E24	E matrix element
45-48	4	E31	E matrix element
49-52	4	E32	E matrix element
53-56	4	E33	E matrix element
57-60	4	E34	E matrix element
61-64	4	E41	E matrix element

Bytes	Total Bytes	Parameter	Explanation
1-2	2	PacketType	45,1
3-4	2	Tag	3 (left eye) or 6 (right eye)
5-8	4	E42	E matrix element
9-12	4	E43	E matrix element
13-16	4	E44	E matrix element
17-20	4	Rpc	The distance in camera pixels between centre of corneal curvature and centre of eye rotation
21-24	4	X_singleGlintOffset	X distance from single glint in camera pixels, to centre of corneal curvature
25-28	4	Y_singleGlintOffset	Y distance from single glint in camera pixels, to centre of corneal curvature
29-64	4	Unused	

Appendix B: CDC Commands

System Information (read only) commands

>>$Help

This will return a list of possible commands. See the rest of this document for explanations for what each does.

>> $ProductType

Returns the productType in the format:

$ProductType;LiveTrack-FM;

>> $SerialNumber

This returns the 8 character serial number of the device in the following format (12345678 will be replaced by the serial number specific to your device):

$SerialNumber;12345678;

>> $FirmwareDate

This will return the date the current firmware was compiled. This is of little use to the user however if there is a problem and CRS need to be contacted for support, including the date of the current firmware may help to resolve the problem quicker. The returned string will be in the format:

$FirmwareDate;DD/MM/YYYY hr:min;

Commands Relating to Data Acquisition

>>$ArmTimestamp=[ON]

The next rising edge applied to the trigger input will zero the frame counter. This is used to synchronise the eye-tracking with typically the visual stimulus.

>> $Left = [ON]

>> $Left = [OFF]

Turns tracking of the left eye on or off respectively.

>> $Right = [ON]

>> $Right = [OFF]

Turns tracking of the right eye on or off respectively.

>> $Raw

Starts returning uncalibrated data on pupil and glint positions and sizes, in camera pixel units. This will continue until the $Stop command is given (see below). Returned data also includes which eye the data refers to, the timestamp of the measurement and whether a trigger was

received by the LiveTrack system for that frame (triggerIn equals 0 if no trigger or 1 if trigger present). An example format of the returned data for the left eye is:

$leftRaw;*timeStamp*;*triggerIn*;*pupilX*;*pupilY*;*pupilDiameterX*;pupilDiameterY;*glint1X*;*glint1Y*;*glint2X*;*glint2Y*;

Note that if the eye was not tracked for that frame, the pupil area will be 0. This is the only reason the pupil area should be 0 and so can be used to separate frames which were tracked or those which were not tracked.

>> $Calibrated

Device starts reporting tracking data. This is similar to the $Raw command except this command uses calibration data to return information on gaze position rather than raw data. It will use the default calibration parameters from the xml file unless new values have been uploaded. The returned string also specifies which eye the data relates to, whether a trigger was received (triggerIn = 1) or not (triggerIn = 0) and the camera frame count for the data. An example of the format of the returned data for the left eye is:

$leftEye;*timeStamp*;*triggerIn*;*calibratedEyeX*;*calibratedEyeY;calibratedEyeZ*;*pupilArea*;

Note that if the eye was not tracked successfully for that frame, the pupilArea will be 0. This is the only reason the pupilArea should be 0 and so can be used to sort frames which were successfully tracked or not.

>>$Fick

Similar to above, but returns angular gaze results in Fick format.

$leftEye;*timeStamp*;*triggerIn*;*Latitude*;*Longitude*;*pupilArea*;

>>$Helmholtz

Similar to above, but returns angular gaze results in Helmholtz format.

$leftEye;*timeStamp*;*triggerIn*;*Azimuth*;*Elevation*;*pupilArea*;

>> $Stop

Stops recording tracking data and re-enters the ‘ready state’.

>> $VideoOverlay = [0]

>> $VideoOverlay = [1]

>> $VideoOverlay = [2]

>> $VideoOverlay = [3]

Turns the video overlay on or off for pupil, glint and frame counter. On the video feed of the eye tracking, it is possible to have the estimated ellipses which are being fit to the pupil and glints to be displayed or not. Displaying them can help to check that the fits look reasonable, that it is not attempting to track something other than the pupil or corneal reflections (glints), such as reflections from glasses or the black of mascara. However for some purposes, it may be preferable to have just the image of the eye itself, without the ellipses.

Commands related to calibration

>> $CalAMatrixLeft

>> $CalAMatrixRight

Return the calibration A matrices (4x4) currently stored and used for the left or right eye respectively. An example of the format that data are returned in for the left eye is:

$CalAMatrixLeft= [*A11 A12 A13 A14*; *A21 A22 A23 A24*; *A31 A32 A33 A34*; *A41 A42 A43 A44*]

>> $CalAMatrixLeft=
[*A11;A12;A13;A14;A21;A22;A23;A24;A31;A32;A33;A34;A41;A42;A43;A44*]

>> $CalAMatrixRight=
[*A11;A12;A13;A14;A21;A22;A23;A24;A31;A32;A33;A34;A41;A42;A43;A44*]

These commands can be used to upload a new calibration A matrix for the left or right eye respectively.

>> $CalRPCLeft

>> $CalRPCRight

Specify new values in camera pixel units, for the distance between the centre of corneal curvature, and the centre of eye rotation. These values do not have to be very accurate as the calibration procedure corrects for errors. However it is best to err on the large side for these values.

>> $PixelsToMM

Used to calibrate the pupil area. Scaling factor between camera pixels and chosen real-world dimensional units (e.g. mm).