

ViewPoint EyeTracker® for BIOPAC Eye Tracking Systems

PC-60 Software Users Guide

BIOPAC Eye Tracking Systems	Monocular	Binocular
Eye Track Sys: Fixed Head+Clamp	EYEFIXMONOCLAMP	EYEFIXBINOCLAMP
Eye Track with HMD1	EYETRAKHMD1MONO	EYETRAKHMD1BINO
Eye Track for HMD2	EYETRAKHMD2MONO	EYETRAKHMD2BINO
Eye Track for 3rd-Party HMD	EYETRAKHMD3RDMO	EYETRAKHMD3RDBI
Eye Track System: Fixed Head	EYEFIXMONO	EYEFIXBINO
Eye Frame Mounted Scene	EYEFramescENEMO	EYEFramescENEBI

New users should study Chapters 2, 3 and 4 to get started:

- [Chapter 2: Overview](#) of *ViewPoint* Describes the theory and provides an overview of the design of the *ViewPoint EyeTracker*®.
- [Chapter 3 Installation and Setup](#) of the video capture hardware and driver installation process and *ViewPoint EyeTracker*® software installation.
- [Chapter 4 Tutorial](#) A brief tutorial designed to help new users start recording eye movements very quickly and easily.

The most comprehensive reference section is Chapter 14 ViewPoint Interface: GUI, SDK, CLP

- **This chapter covers all aspects of the eye tracker and should be referred to often.**

The remaining chapters each deal with a different task or set of tasks that the user will want to perform and provide some helpful background to eye tracking.

Use menu item: **Help > Documentation** ... to quickly access the Documentation folder.

Contact Information

Please visit our website frequently for updates and new product announcements.
www.biopac.com > [eye tracking](#) or send email to: support@biopac.com.

10 October 2005 1:19 PM

ViewPoint EyeTracker® is a registered trademark of Arrington Research, Inc.
ViewPoint ~ Voltage™ is a trademark of Arrington Research, Inc.
RemoteLink™ is a trademark of Arrington Research, Inc.



<u>CHAPTER 1</u>	<u>INTRODUCTION.....</u>	<u>1</u>
1.1	Congratulations	1
1.2	Custom Software & Hardware Development	1
1.3	User Feedback	1
1.4	License Information and Conditions of Use	2
1.5	High-Risk Activities Warning.....	2
1.6	Special Thanks	2
1.7	How to Use this User Guide	2
1.8	Info Window.....	3
1.9	Support.....	3
1.10	Citing <i>ViewPoint</i>	3
<u>CHAPTER 2</u>	<u>OVERVIEW OF <i>VIEWPOINT</i>.....</u>	<u>4</u>
2.1	General Description.....	4
2.2	Infrared Light	6
2.3	Mapping to Gaze Point	6
2.4	Software Developer's Kit (SDK).....	7
2.5	Command Line Parser (CLP)	9
2.6	Inter-Computer & Inter-Application Communication	9
<u>CHAPTER 3</u>	<u>INSTALLATION AND SETUP</u>	<u>11</u>
3.1	Computer System Requirements.....	11
3.2	Video Requirements	11
3.3	Using with Third Party Video Input Equipment	11
3.4	PCI Video Capture Card and Driver Installation	12
3.5	ViewPoint EyeTracker [®] Software Installation.....	14
3.6	License Agreement.....	14
3.7	ViewPoint License (.VPL) File	14
3.8	Menu Navigation.....	15
3.9	User Windows	16
<u>CHAPTER 4</u>	<u>TUTORIAL</u>	<u>18</u>
4.1	Stimulus Window Positioning (Head Fixed).....	18
4.2	EyeCamera Window Setup.....	18

4.3	Corrective Lenses (Eye Glasses)	19
4.4	Thresholding	19
4.5	Calibration (Head Fixed)	22
4.6	Data Collection and Analysis	23
4.7	Sampling Rate	23
4.8	Frequently Used Settings	23
4.9	Preferred Window Layout	24
4.10	Accelerator Keys	24
4.11	Printing	24
 <u>CHAPTER 5 LOCATING THE PUPIL AND GLINT</u>		<u>25</u>
5.1	Feature Method	25
5.2	Noise	26
5.3	Automatic Slip Compensation	26
5.4	Simulation of Gaze	27
5.4.1	Manual Simulation	27
5.4.2	Pattern Simulation (<i>only on special versions</i>)	27
5.5	Setting the Search Regions	27
5.6	Brightness and Contrast Adjustments	28
5.6.1	Video AutoImage	28
5.7	Thresholding and Setting the Scan Density	28
5.7.1	Setting the Scan Density	29
5.7.2	AutoThreshold	29
5.7.3	Positive Lock Threshold Tracking	29
5.8	Criteria	29
5.8.1	Pupil Aspect Criterion	30
5.8.2	Width Criteria	30
5.9	Step-by-step guide for <i>Glint-Pupil Vector</i> method	30
5.10	Alternative Segmentation Methods	31
5.10.1	Centroid	32
5.10.2	Edge Trace (<i>only on special versions</i>)	32
5.10.3	Oval Fit	32
5.10.4	Ellipse (<i>only on special versions</i>)	32
5.10.5	Glint Segmentation Methods	32
5.11	Pupil Scan Area Shape Options	32
 <u>CHAPTER 6 CALIBRATION</u>		<u>33</u>
6.1	Raw Data Without Calibration	33
6.2	Calibration Description	33
6.3	Calibration Carryover	33
6.4	Calibration Procedure (Head Fixed)	33

6.5	Assessing Calibration Success.....	34
6.6	Omitting Individual Calibration Points.....	35
6.7	Re-presenting Individual Calibration Data Points	36
6.8	Slip Correction	36
6.9	Instructions to Subject	36
6.10	Dominant Eye	36
6.11	Saving Calibration Eye Images.....	37
6.12	Advanced Calibration Controls	37
6.12.1	Presentation Order	37
6.12.2	Snap and Increment Calibration Modes	37
6.12.3	Adjusting the Calibration Area	38
6.13	Custom Calibration Point Positions	38
6.14	Geometry Grid	38
 <u>CHAPTER 7</u> <u>CURSOR CONTROL</u>		<u>40</u>
 <u>CHAPTER 8</u> <u>OCULAR TORSION.....</u>		<u>41</u>
8.1	Introduction to Torsion.....	41
8.2	Procedure for Measuring Torsion	42
8.3	Torsion Demonstration Test	43
8.4	Overriding the Default Torsion Parameters	44
 <u>CHAPTER 9</u> <u>STIMULUS PRESENTATION (HEAD FIXED).....</u>		<u>45</u>
9.1	General.....	45
9.2	Picture Lists.....	45
9.3	Using the Stimulus Window (Head Fixed Option).....	47
9.4	Using the GazeSpace Window	47
9.5	Regions of Interest (ROI).....	48
9.6	Data Smoothing.....	49
9.7	Using the SDK, settings files and Serial Port Interface for Stimulus Presentation	49
9.8	Integrating with Third Party Products.....	50
 <u>CHAPTER 10</u> <u>DATA COLLECTION</u>		<u>51</u>
10.1	Sampling Rate	51
10.2	Saving Data to File	51
10.3	Data File Format.....	52
10.3.1	File header information.....	52
10.3.2	File records.....	52

10.3.3	Synchronous vs. Asynchronous data inserts.....	52
10.3.4	Data Record Tags.....	53
10.4	Direction-of-gaze Coordinates.....	56
10.5	Timing Measurement.....	57
10.6	Region of Interest (ROI).....	57
10.7	Quality Marker Codes.....	57
10.8	Pupil Diameter.....	58
10.9	Pupil Aspect.....	59
10.10	Display Screen Geometry.....	59
 <u>CHAPTER 11 DATA ANALYSIS.....</u>		60
11.1	Real-Time.....	60
11.1.1	Data Smoothing.....	61
11.2	Fixation, Saccade, Drift and Blinks.....	61
11.2.1	Velocity Threshold.....	61
11.2.2	Fixations.....	61
11.2.3	Drift.....	62
11.2.4	Blinks.....	62
11.2.5	Events.....	62
11.2.6	SDK.....	63
11.3	Post-Hoc.....	63
 <u>CHAPTER 12 USING SETTINGS FILES.....</u>		64
12.1	CLP String Parsing.....	64
12.2	Saving and Loading Settings Files.....	64
12.3	Pre-load Settings in a Startup file.....	65
12.4	Settings/LastRun.txt.....	65
12.5	Settings File Lists.....	65
12.6	SettingsFile Examples.....	65
12.7	CLP Commands.....	66
12.8	Associating CLP Commands with FKeys.....	66
 <u>CHAPTER 13 SERIAL PORT COMMUNICATION.....</u>		67
13.1	Getting Started.....	67
13.2	Sending Real and Test Data.....	67
13.3	Transfer to Intel and Macintosh Machines.....	68
13.4	Connections.....	68
13.5	Serial Protocol.....	68
13.6	Serial Packet Header Structure.....	69

13.7	Serial Packet Data Structures.....	70
13.8	Data Value Encoding.....	71
13.9	Packet Data Structures.....	72
13.10	Example Serial Port Code	72
<u>CHAPTER 14 VIEWPOINT INTERFACE: GUI, SDK, CLP</u>		74
14.1	General.....	74
14.1.1	VPX_SendCommand("setSomething") replaces VPX_SetSomething	74
14.1.2	VPX_SendCommand & formatted strings	74
14.1.3	Quoting strings with white spaces	75
14.1.4	Case insensitive CLP strings.....	75
14.1.5	Boolean Toggle	75
14.1.6	SDK return values	75
14.2	Data Files	76
14.2.1	Open Data File with Randomly Generated Name	76
14.2.2	Specify NewUnique Data File Extension	76
14.2.3	Open a Data File and Specify a File Name	77
14.2.4	Insert a String into the Data File	77
14.2.5	Insert a Marker into the Data File	78
14.2.6	Insert a User Defined Data Tag into the Data File.....	79
14.2.7	Specifies asynchronous or synchronous string data	79
14.2.8	Specify asynchronous or synchronous markers data	80
14.2.9	Specify asynchronous or synchronous head tracker data	80
14.2.10	Specify data file start time.....	81
14.2.11	Store smoothed or unsmoothed data.....	82
14.2.12	Specify whether to use buffering (DEPRECATED).....	82
14.2.13	Pause writing of data to file.....	83
14.2.14	Close Data File	83
14.2.15	Close Data File and Open in Post-Hoc Analysis tool	84
14.3	Stimulus Images	84
14.3.1	Load Stimulus Image into the Stimulus window	84
14.3.2	Specifies how to display the currently loaded stimulus image	85
14.3.3	Specify a background "matting" color for the stimulus window	85
14.3.4	Play specified Sound file	86
14.4	PictureList.....	86
14.4.1	Initialize Picture List.....	86
14.4.2	Add List of Image Names to PictureList.....	86
14.4.3	Randomize List of Images in the PictureList.....	87
14.4.4	Move to Next Image in the PictureList.....	87
14.4.5	Move to Start of Images in Picture List.....	87
14.5	Controls window: VideoImage	88

14.5.1	Specify Mapping Feature.....	88
14.5.2	AutoThreshold	88
14.5.3	Positive Lock Tracking.....	89
14.5.4	Adjust Pupil Threshold Slider	89
14.5.5	Adjust Glint Threshold Slider	90
14.5.6	Adjust Video Image Brightness	90
14.5.7	Adjust Video Image Contrast.....	91
14.5.8	Dynamically Optimize Brightness and Contrast Settings.....	91
14.5.9	Adjust Pupil Scan Density	92
14.5.10	Override Pupil Scan Density Minimum.....	93
14.5.11	Adjust Glint Scan Density	94
14.5.12	Override Glint Scan Density Minimum	95
14.6	EyeCamera Window	95
14.6.1	Adjust Pupil Sacn Area.....	95
14.6.2	Specify Pupil Scan Area Shape.....	96
14.6.3	Pupil and Glint oval fit constraints	96
14.6.4	Define Glint Scan Area	96
14.6.5	Define Offset of Glint Sacn Area Relative to the Pupil.....	97
14.6.6	Unyoke Glint Scan Area from the Pupil	97
14.6.7	Define offset of Unyoked Glint Scan Area	98
14.6.8	Toggle Show Treshold Dots On / Off.....	98
14.6.9	Specify EyeImage Overlay Graphics sent to layered application (EXPERIMENTAL).....	99
14.6.10	EyeCamera Tool Bar Display	99
14.7	Video related controls.....	100
14.7.1	Specify Video Input Standard	100
14.7.2	Specify Tracking Operation Mode	100
14.7.3	Specify Dark or Bright Pupil Tracking	101
14.7.4	Specify Pupil Segmentation Method.....	101
14.7.5	Specify Glint Segmentation Method	101
14.7.6	Changes default setting for Freeze Feature	102
14.7.7	Toggle Freeze Video Imge Preview On / Off.....	102
14.7.8	Reset Video Capture Device	102
14.8	Calibration controls.....	103
14.8.1	Start Auto-Calibration	103
14.8.2	Stop Auto-Calibration	103
14.8.3	Specify Calibration Stimulus Presentation Speed	103
14.8.4	Specify the duration of presentation of calibration warning notice 104	
14.8.5	Specifies Interval Between Presentation of Calibration Stimulus Points 104	
14.8.6	Calibration Snap Mode	105
14.8.7	RePresent in Snap Calibration Mode	105
14.8.8	AutoIncrement Calibration Mode	106

14.8.9	Calibration Stimulus Point Presentation Order	106
14.8.10	Specify Number of Calibration Stimulus Points.....	106
14.8.11	Specify Calibration Stimulus Point Color.....	107
14.8.12	Specify Calibration Stimulus Window Background Color	107
14.8.13	Randomize Calibration Stimulus Points Check Box (DEPRECATED).....	107
14.8.14	Specify Calibration Stimulus Point Presentation Order	108
14.8.15	Specify Individual Custom Calibration Stimulus Points	108
14.8.16	Display Custom Calibration Stimulus Point Order.....	109
14.8.17	Select the Specified Calibration Data Point	109
14.8.18	Select the Index Number that Maps to the Specified Calibration Data Point	110
14.8.19	Undo the last operation on a Calibration Data Point	110
14.8.20	Re-Present the Specified Calibration Data Point	111
14.8.21	Specify Custom Calibration Stimulus Point Locations.....	111
14.8.22	Turn Custom Calibration Stimulus Point Location ON / OFF ...	112
14.8.23	Print Locations of custom calibration stimulus points in EventHistory window	112
14.8.24	Controls display of nearest-neighbor gridlines in the EyeSpace window	112
14.8.25	Compensate for Slip	113
14.8.26	Adjust Calibration Area	113
14.8.27	Save Image of Eye at each Calibration Data Point	113
14.9	Controls: Criteria Controls	114
14.9.1	Specify amount of Smoothing.....	114
14.9.2	Specify Smoothing Algorithm to Apply.....	114
14.9.3	Specify Velocity Threshold	115
14.9.4	Specify amount of Drift Allowed.....	115
14.9.5	Specify Pupil Aspect Ratio Failure Criterion	116
14.9.6	Specify Pupil Width Failure Criterion	116
14.10	Region of Interest (ROI)	117
14.10.1	Define an ROI Box	117
14.10.2	Specify Number of ROI to be drawn in a circle around center of window	117
14.10.3	Remove all ROI Boxes.....	118
14.10.4	Select a Specific ROI	118
14.10.5	Select the next ROI Box.....	118
14.10.6	Lock ROI Settings	119
14.11	PenPlot controls	119
14.11.1	Specify Which PenPlot Traces to Display	119
14.11.2	BackGround Color of PenPlot Traces	120
14.11.3	PenPlot Back Ground Color	120
14.11.4	Specify Speed of PenPlot Scrolling.....	120
14.11.5	Specify Range of PenPlot Values	121

14.11.6	Specify the behavior of the penpot after a video freeze	121
14.12	Graphics controls.....	122
14.12.1	Specify the color of the GazeSpace and PenPlot Lines.....	122
14.12.2	Specify which Overlay Graphics to Display in the GazeSpace Window	122
14.12.3	Specify which Overlay Graphics to Display in the Stimulus Window	123
14.12.4	Erase Data Displays in the GazeSpace and Stimulus windows	123
14.12.5	Automatically erase display windows.....	124
14.12.6	Specify time delay for auto erase.....	124
14.13	Stimulus Window controls	125
14.13.1	Specify Stimulus Source	125
14.13.2	Specify Custom Stimulus window Size and Position.....	126
14.13.3	Automatically Show the Stimulus Window on Primary Monitor	127
14.13.4	Specify How and where to show Stimulus window	128
14.13.5	Calibrate to a third party application window.....	129
14.14	Window related controls	129
14.14.1	Print ViewPoint windows.....	129
14.14.2	Include Date and Time Stamp on Printed windows.....	130
14.14.3	Move and Resize Window	130
14.14.4	Specify ViewPoint Window Layout.....	131
14.14.5	Clear Event History window	131
14.14.6	Save window layout settings	132
14.15	Settings File commands	132
14.15.1	Load Settings File	132
14.15.2	Edit Settings File	132
14.15.3	Show Verbose Settings File Loading Details in Event History	133
14.15.4	Save Settings e.g. calibrations etc.....	133
14.16	SettingsFileList commands.....	133
14.16.1	Initialize Settings File List.....	133
14.16.2	Next Settings File in List	134
14.16.3	Add Settings File to the List	134
14.16.4	Restart Settings File List.....	134
14.16.5	Toggle Autosequencer ON / OFF	134
14.16.6	Specify delay between Settings Files in List	135
14.17	Torsion commands	135
14.17.1	Start / Stop Torsion Calculations.....	135
14.17.2	Adjust Start Point of Torsion Sampling Arc.....	136
14.17.3	Adjust Radius of Torsion Sampling Arc.....	136
14.17.4	Autoset Torsion Template after Adjustments	137
14.17.5	Display Real-Time Torsion Data	137
14.17.6	Adjust Torsion Measurement Range.....	138
14.17.7	Adjust Torsion Measurement Resolution	138

14.17.8	Set Autocorrelation Template	139
14.18	Interface settings commands.....	139
14.18.1	Turn Cursor Control On / Off.....	139
14.18.2	Use Fixation to Issue Button Click	139
14.18.3	Specify Fixation Time to Issue Button Click.....	140
14.18.4	Use Blinks to Issue Button Click	140
14.19	RemoteLink & SerialPort controls.....	140
14.19.1	Connect / Disconnect Serial Port	140
14.19.2	Specify Serial Data to Send	141
14.19.3	Send Serial Port Ping.....	141
14.20	HeadTracking commands.....	141
14.20.1	Connect / Disconnect Head Tracker Interface	141
14.20.2	Specify whether to use local or global origin	142
14.20.3	Reset Origin for the Head Sensor	142
14.20.4	Set Position and Angle Origins	142
14.20.5	Specify the Vector between Head Sensor and the Eyeball.....	143
14.20.6	Turn CRT pulse synchronization On / Off	143
14.21	Binocular commands	143
14.21.1	Turn Binocular Mode On / Off	143
14.21.2	Specifies Binocular Averaging	144
14.22	File Related	144
14.22.1	Launch ViewPoint with Command Line Options	144
14.22.2	Specify Default ViewPoint Folder path.....	145
14.23	FKey	146
14.23.1	Associate CLP Commands with FKeys.....	146
14.24	TTL	146
14.24.1	Associate CLP Commands with TTL Voltage Changes	146
14.24.2	Set TTL Output Voltages	147
14.24.3	Simulate Change in TTL Input	147
14.24.4	Set TTL Output to Indicate Data Quality Codes	148
14.25	Misc	149
14.25.1	Specify Verbose Information to Send to History Window	149
14.25.2	Update Eye Data on Request	150
14.25.3	Set Status Window Update Rate for FPS Field.....	150
14.25.4	SDK Debug Mode	150
14.25.5	Specify ViewPoint Generated Events	151
14.25.6	Turn Accelerator Key Functionality On / Off.....	151
14.26	Parser Instructions.....	151
14.26.1	Settings File Comment.....	151
14.26.2	End of Settings File Command	152

CHAPTER 15 SOFTWARE DEVELOPERS KIT (SDK) 153

15.1	General.....	153
15.2	Registering to Receive Notifications	153
15.3	Example SDK Code.....	155
15.4	Data Quality Codes	155
15.5	Sending CLP Commands with the SDK.....	156
15.6	High Precision Timing.....	156
15.7	DLL Version Checking.....	156
15.8	SDK Access Functions	156
15.8.1	Get Eye Data Access	157
15.8.2	Get Time Information.....	163
15.8.3	Get Motor Data	164
15.8.4	Get ViewPoint Status.....	165
15.8.5	Get ViewPoint Stimulus Window	166
15.8.6	Get Stimulus Display Geometry	167
15.8.7	Get ROI	167
15.8.8	Set Remote EyeImage	169
15.9	DLL Interface	171
15.10	ViewPoint Events & Notification Messages	173
15.10.1	General Events	173
15.10.2	Calibration Events.....	174
 <u>CHAPTER 16</u> <u>LEGACY, OBSOLETE, & DEPRECATED</u>.....		179
16.1.1	Old CLP.....	179
16.1.2	Old VPX.....	180
 <u>CHAPTER 17</u> <u>TROUBLESHOOTING</u>		181
17.1	EventHistory Window.....	181
17.2	Improving Frame Rate.....	181
17.3	EyeCamera Window Troubleshooting	181
17.3.1	Bottom half of EyeCamera window is black.....	182
17.4	General Troubleshooting	182
 <u>CHAPTER 18</u> <u>HISTORY OF EYE TRACKING METHODS</u>		183
18.1	Electrical Methods	183
18.1.1	Surface Recordings	183
18.1.2	Induction Coils.....	183
18.2	Optical Methods.....	183
18.2.1	Reflections, or Purkinje Images.....	183
18.2.2	Dark Pupil Tracking	184



18.2.3	Limbus Tracker.....	184
18.2.4	Bright Pupil Method	184
18.2.5	Corneal Bulge Method.....	184
18.2.6	Vector Difference Method	184
<u>CHAPTER 19</u> <u>BINOCULAR OPTION</u>		<u>186</u>
19.1	Installing Binocular FrameGrabber & Software.....	186
19.2	Operating in Binocular Mode	186
19.3	Setup	186
19.4	Storing Data.....	186
19.5	Real-Time Display of Binocular Data	187
19.6	Interfacing to Other Applications.....	187
19.7	Settings Files	188
<u>CHAPTER 20</u> <u>HEAD TRACKER OPTION</u>		<u>189</u>
20.1	General.....	189
20.2	Unpacking.....	189
20.3	Installation	189
20.4	Cable Attachment.....	189
20.5	Baud Rate and DipSwitch Settings	190
20.6	Connection to FOB	190
20.7	CRT Synchronization.....	191
20.8	Location and Orientation of Transmitter & Receiver	191
20.9	Head Tracker Event Data	192
20.10	HeadSpace Window	192
20.11	Troubleshooting.....	192
<u>ARI SOFTWARE LICENSE</u>		<u>193</u>
<u>THIRD PARTY LICENSES</u>		<u>195</u>

Chapter 1 Introduction

1.1 Congratulations

Congratulations on your purchase of the *ViewPoint EyeTracker*[®]. It has been designed to be the easiest to use, most reliable and best value eye tracker on the market. Because the solution is primarily software, it has several advantages:

- No expensive and cumbersome hardware to configure and maintain.
- Easy integration with other application programs.
- Standard user interface panels.
- Upgrades are trivial to install and relatively inexpensive.
- Performance will increase when the computer is upgraded.

It provides:

- A comprehensive solution for eye tracking research.

An embeddable eye tracking solution for 3rd party products and end user custom applications.

The *ViewPoint EyeTracker*[®] was originally developed in 1995 for the Apple Macintosh platform. It has now been ported to the Microsoft Windows platform. It is our intention to make both versions essentially identical. However, product development cycles mean that new features will sometimes appear on one platform before appearing on the other. Please visit our web site regularly for further hardware and software developments and platform specific variations

1.2 Custom Software & Hardware Development

Special software and hardware development for particular laboratories or organizations may be performed under individual consulting agreements. *ViewPoint EyeTracker*[®] is easily customized as an embedded eye tracking solution for OEMs. We also help OEMs to interface the *ViewPoint EyeTracker*[®] with their equipment by providing custom camera and optical solutions. Please email inquiries to: info@ArringtonResearch.com

1.3 User Feedback

Suggestions for improvements to this manual or to the *ViewPoint EyeTracker*[®] software are always welcome and appreciated. Please email comments to: info@ArringtonResearch.com

1.4 License Information and Conditions of Use

Use of the software constitutes consent to the terms of the “ARI Software License” on page 193. The contents of this user guide and any other documentation provided with the *ViewPoint EyeTracker*® is for the use of registered *ViewPoint EyeTracker*® users only. No part of this or other *ViewPoint EyeTracker*® documentation or Software Developer Kit (SDK) information may be distributed or shared with others, without prior written permission from Arrington Research, Inc.

1.5 High-Risk Activities Warning

Every effort has been made to provide a bug-free product. Nevertheless, this software is not intended for use in the operation of nuclear facilities, aircraft navigation or communications systems, or air traffic control, or medical treatment and diagnosis, or for any other use where the failure of the software could lead to death, personal injury, damage to property or severe environmental damage.

1.6 Special Thanks

The initial stages of the *ViewPoint EyeTracker*® project were greatly facilitated by the generosity of Professor Richard Held, M.I.T., Dr. Yasuo Nagasaka, Rikkyo University; many thanks and deep gratitude is given to them.

1.7 How to Use this User Guide

New users should study Chapters 2, 3 and 4 to get started:

Chapter 2: Overview of ViewPoint Describes the theory and provides an overview of the design of the *ViewPoint EyeTracker*®.

Chapter 3 Installation and Setup of the video capture hardware and driver installation process and *ViewPoint EyeTracker*® software installation.

Chapter 4 Tutorial A brief tutorial designed to help new users start recording eye movements very quickly and easily.

Chapter 14 ViewPoint Interface: GUI, SDK, CLP is the most comprehensive reference section for all aspects of the eye tracker and should be referred to often.

The remaining chapters each deal with a different task or set of tasks that the user will want to perform and provide some helpful background to eye tracking.

Use menu item: [Help > Documentation ...](#) to quickly access the Documentation folder.

Type fonts are used with the following meanings:

Table 1: Meaning of Type Fonts	
Type Font	Example Meaning
Eye tracking has many applications.	Normal text
<code>stimulusWindowDimensions</code>	Program variable, or SDK code
Load PICT image	User selection: Menu Item or Button
Video window	Program Window
<u>Figure 1:</u>	Link to section, figure or table

1.8 Info Window

The **Info** window can be displayed using menu item **Help > Info**. This provides system information, a list of keyboard shortcut keys, display devices that *ViewPoint* has detected, calibration mapping precision information, etc.

1.9 Support

For support questions send email to: Support@ArringtonResearch.com

1.10 Citing *ViewPoint*

Please use the following format when citing the *ViewPoint EyeTracker*[®].

***ViewPoint EyeTracker*[®] by Arrington Research, Inc.**

Note that *ViewPoint* is one word with only the letters V and P capitalized, and that *EyeTracker* is also one word, with only the letters E and T capitalized. It is a registered trademark and should be followed by a capital R within a circle, ®, or if not available, a capital R within parentheses, (R).

You may also wish to include the web site address (www.ArringtonResearch.com), where ArringtonResearch is one word. Correct capitalization is not required for the web link to work properly, however using only a capital A and a capital R is the preferred form and it makes the link more readable.

We love to hear about your research, published or not, please let us know what you are working on!

Chapter 2 Overview of *ViewPoint*

2.1 General Description

The *ViewPoint EyeTracker*[®] provides a complete eye movement evaluation environment including integrated stimulus presentation, simultaneous eye movement and pupil diameter monitoring, and a Software Developer's Kit (SDK) for communicating with other applications. It incorporates several methods that a user can select from to optimize the system for a particular application. It provides several methods of mapping position signals extracted from the segmented video image in *EyeSpace*[™] coordinates to the participant's point of regard in *GazeSpace*[™] coordinates.

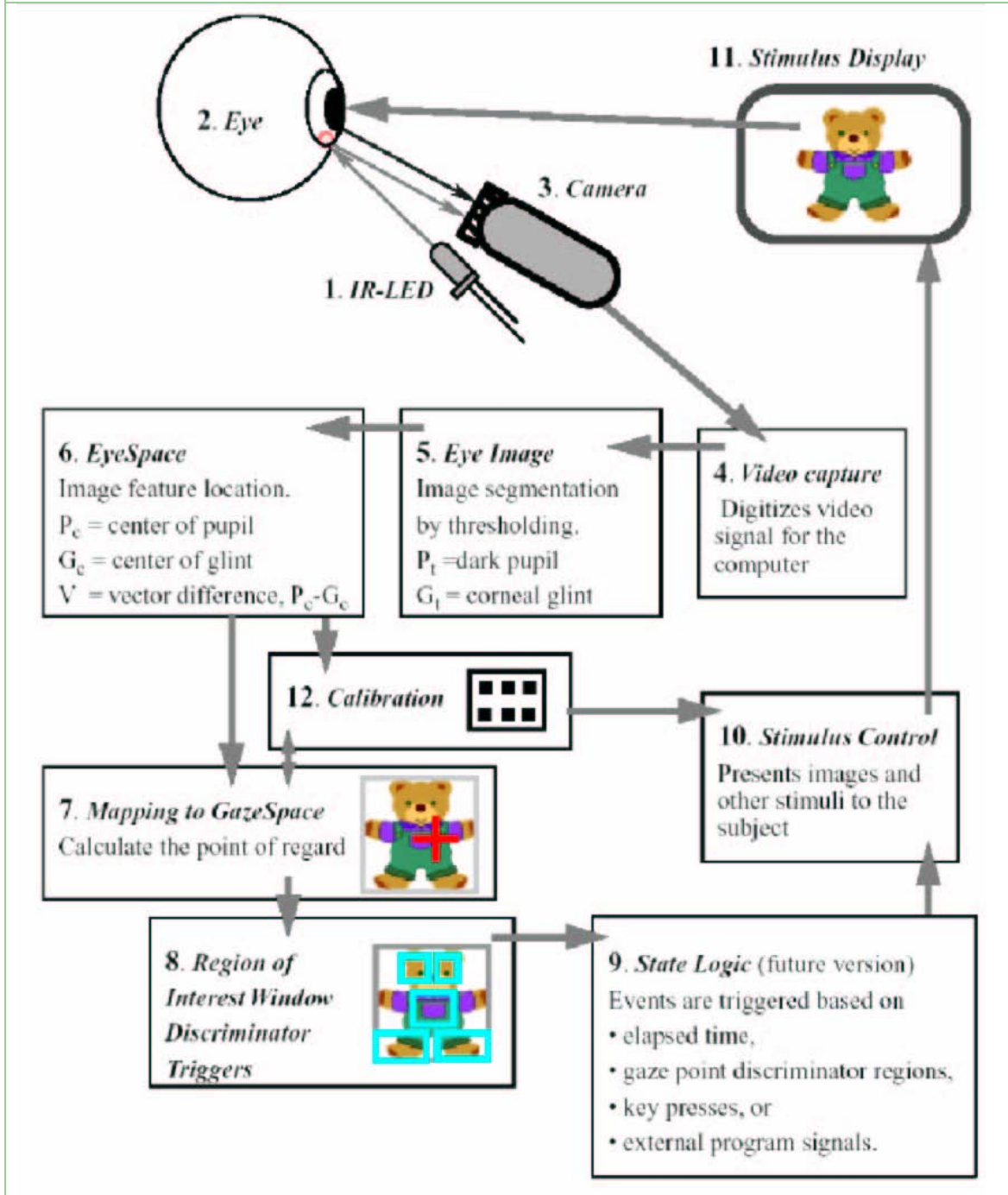
Figure 1: Shows how the *ViewPoint EyeTracker*[®] works in a typical head fixed configuration. The numbers in this section refer to the item or block numbers in the figure.

The infrared light source (item **1.**) serves to both illuminate the eye (item **2.**) and also to provide a specular reflection from the surface of the eye, i.e., from the smooth cornea. In dark pupil mode, the pupil acts as an infrared sink that appears as a black hole; see Figure 2:. In bright pupil mode, the "red eye" effect causes the pupil to appear brighter than the iris. (Note that a different camera and illuminator configuration is required for bright pupil operation.)

The video signal from the camera (item **3.**) is digitized by the video capture device (item **4.**) into a form that can be understood by a computer. The computer takes the digitized image and applies image segmentation algorithms (item **5.**) to locate the areas of pupil and the bright corneal reflection (glint). Additional image processing (item **6.**) locates the centers of these areas and also calculates the difference vector between the center locations. A mapping function (item **7.**) transforms the eye position signals (item **6.**) in *EyeSpace* coordinates to the subject's *GazeSpace* coordinates. (Item **8.**) Next, the program tests to determine whether the gaze point is inside of any of the region of interest (ROI) that the user has defined.

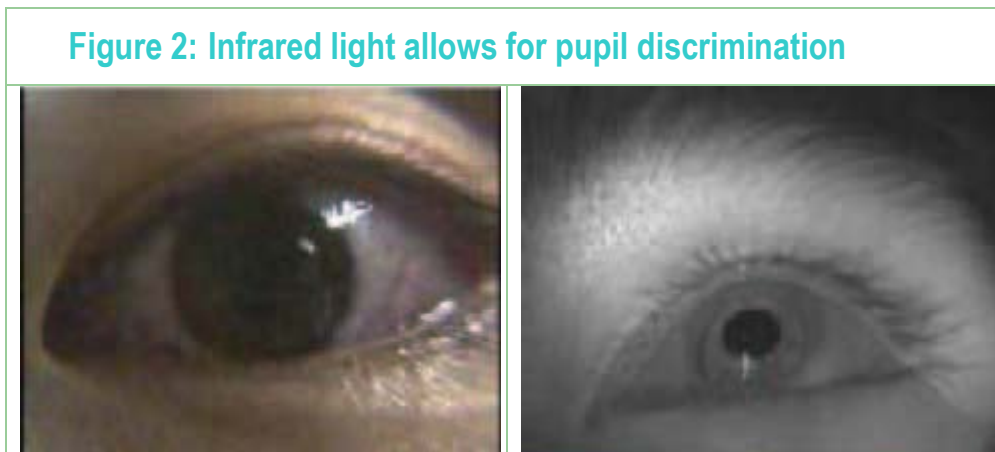
The calibration system (item **12.**) can be used to present calibration stimuli via (item **10.**) to the user and to measure the eye position signals (item **6.**) for each of the stimulus points. These data are then used by (item **12.**) to compute an optimal mapping function for mapping to position of gaze in *GazeSpace* (item **7.**).

Figure 1: Schematic of the *ViewPoint EyeTracker*[®] System (head fixed)



2.2 Infrared Light

The value of using infrared light is illustrated in [Figure 2](#). The left side of the figure shows an image in normal light; in this subject the pupil of the eye is almost impossible to discriminate from the dark iris. The right side of the figure shows an image of the same eye, but viewed with an infrared sensitive camera under infrared lighting conditions; the pupil is easily discriminated. Note that in each case the subject is wearing a contact lens.



There should always be the utmost concern for the safety of the subject. The issue of safe limits of infrared (IR) irradiance is frequently discussed.

10 mW / cm sq is probably the safe maximum figure for corneal exposure over a prolonged period (Clarkson, T.G. 1989, Safety aspects in the use of infrared detection systems, I. J. Electronics, 66, 6, 929-934).

The infrared corneal dose rate experienced out of doors in daylight is of the order of 10^{-3} W / cm². Safe chronic ocular exposure values particularly to the IR-A, probably are of the order of 10^{-2} W / cm² (D.H. Sliney & B.C. Freasier, Applied Optics, 12:1, 1973).

ISO/DIS 10342 (page 7) gives maximum recommended fundus irradiance for use in Ophthalmic Instruments of 120 mW / sq cm but this is for short-term exposure.

All IR-illuminator and camera systems provided by Arrington Research, Inc. are designed to be well within safe limits of exposure.

2.3 Mapping to Gaze Point

It is often necessary to determine where a person is looking, that is, to determine the *gaze point*, also called the *point of regard*. This task is performed by using a mathematical function to *map* the eye position signal in the *EyeSpace* coordinates of the video image to the gaze point in the *GazeSpace* coordinates of the visual stimulus. There are many algorithms that can be used to perform such a mapping and many of them are company proprietary. By far, the best algorithms are non-linear. This is because the eye movements are rotational, i.e., the translation of the eye

position signal that is apparent to the camera is a trigonometric function of the subject's gaze angle. Moreover, the camera angle may provide an oblique line of sight. *ViewPoint EyeTracker*[®] employs one of the most powerful and robust methods available.

2.4 Software Developer's Kit (SDK)

The *ViewPoint EyeTracker*[®] software includes a powerful software developer's kit (SDK) that allows programs to seamlessly interface with *ViewPoint* in real-time. It provides real-time access to all *ViewPoint* data. It provides for calibration stimuli in the user's stimulus window, or the user's application to draw into *ViewPoint*'s Stimulus and GazeSpace windows. It provides complete external control of the *ViewPoint EyeTracker*. The SDK interface is based on shared memory in a dynamic-link library (DLL).

MATLAB[®] (as of version R17) is now designed so that the ViewPoint DLL can be loaded directly into MATLAB, which means that a user can call ViewPoint SDK functions, as easily as calling MATLAB functions.

The SDK software includes:

The DLL library interface,

.h and .lib files

plus sample source code written in C.

This is documented in detail in [Chapter 14 ViewPoint Interface: GUI, SDK, CLP](#)

The SDK is designed to be very easy to use. A complete program interface is shown here below:

```
#include "vpx.h"

main()
{
  for( int ix=0; ix<999 ; ix++ )
  {
    VPX_RealPoint gp;          // a structure with two floats for (x,y) values
    VPX_GetGazePoint( &gp ); // pass by reference
    printf("ViewPoint gaze point: %g, %g ", gp.x, gp.y );
  }
}
```

This very simply program lists the X and Y positions of gaze calculated by *ViewPoint*.

All eye data is available via easy to use, high level, access functions. Here are a few examples:

```
VPX_GetComponentVelocity(&vel);  
VPX_GetFixationSeconds(&secs);  
VPX_GetTorsion(&degrees);
```

Important status items are also available, for example:

```
VPX_GetStatus( VPX_STATUS_DataFileIsPaused )
```

Several data synchronization mechanisms exist, for example:

```
VPX_SendCommand("dataFile_NewUnique");  
VPX_SendCommand("dataFile_InsertMarker %c", theMarker );  
VPX_SendCommand("dataFile_InsertString \"Showing picture of a cat.\" )  
VPX_SendCommand("dataFile_Pause");
```

2.5 Command Line Parser (CLP)

Every graphical user interface (GUI) selection and adjustment that the user makes in *ViewPoint* (e.g., menu item selection, radio button selection, slider value) can be saved in a Settings file, so that they can be loaded again next time the program is run. The control values are stored as single line ASCII commands in the form of a keyword and parameters. When a Settings file is loaded, each line in the file is sent to the *ViewPoint* command line parser (CLP).

These command strings can also be sent to *ViewPoint* from other programs while *ViewPoint* is running, which means that outside, "layered", programs can have complete control of the *ViewPoint EyeTracker*[®]. These command strings can be sent via the software developers kit (SDK) function `VPX_SendCommand("some command string")`. This can be done from programs running on the same machine or from programs running on remote computers via an Inter-Computer Link.

There are more CLP commands than there are GUI controls in *ViewPoint*. For example there are commands to allow fine control of *ViewPoint* operations and behavior. There are also commands for all the GUI controls, for example, to adjust the amount of data smoothing, to display / hide various pen plots, freeze / un-freeze the eye camera video. There are commands to open, pause, resume, and close *ViewPoint* data files. There are commands to insert remote synchronization data into the *ViewPoint* data files. If there is an action that you need to perform but cannot find a GUI control for it, refer to the [Chapter 14 ViewPoint Interface: GUI, SDK, CLP](#) to see if a CLP command exists.

[Chapter 14 ViewPoint Interface: GUI, SDK, CLP](#) provides a complete description of each type of control.

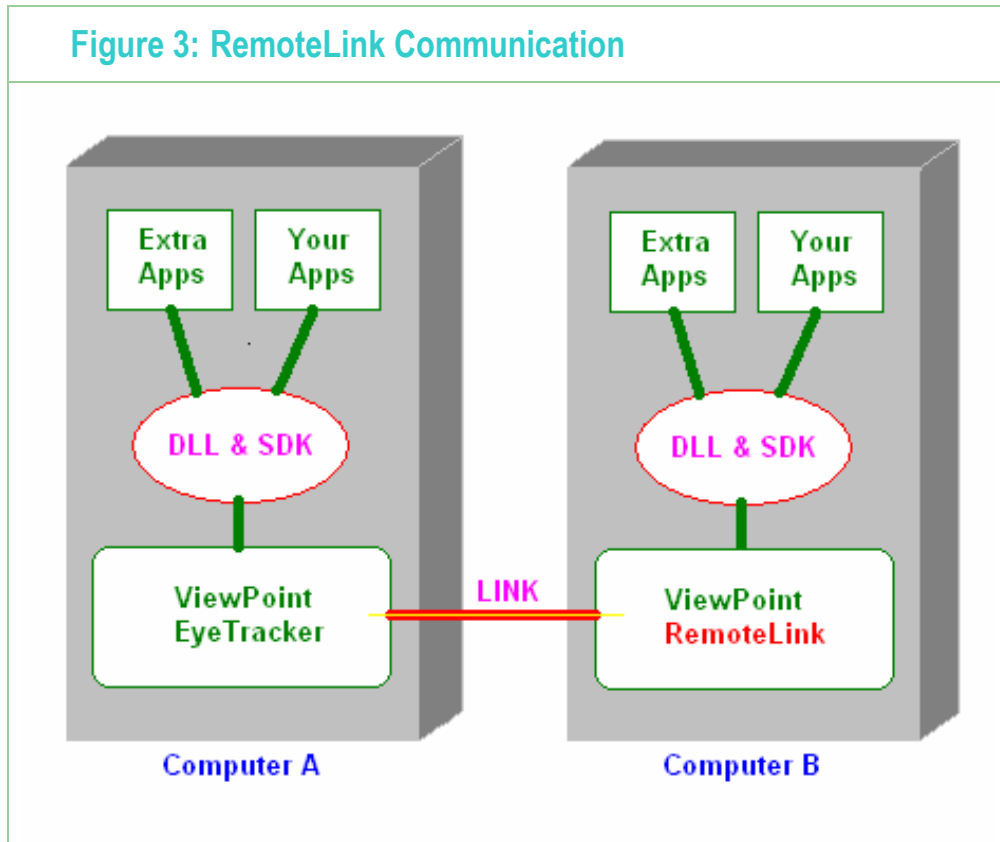
2.6 Inter-Computer & Inter-Application Communication

The *ViewPoint EyeTracker*[®] communicates easily with other applications, either on the same computer or on a remote computer. *ViewPoint* can send eye data and status information to other applications and it can receive instructions from other applications for program control and data synchronization.

The *ViewPoint EyeTracker*[®] software includes the auxiliary program **RemoteLink**[™] to help take care of inter-computer communication. *RemoteLink* can present calibration stimuli on a remote computer's display screen. It can also listen for *ViewPoint* data and update a copy of the DLL on the remote machine, so the application believes that it is talking to *ViewPoint* on the same machine. *RemoteLink* also includes a *ViewPoint* data simulator, for development where *ViewPoint* is not available.

For details about the use of **RemoteLink**[™] refer to the documentation on the *RemoteLink* folder on the disk provided with *ViewPoint*.

This link is currently only via the serial port, however we are developing an Ethernet link that should be available soon. The former is documented in detail later in Chapter 13 Serial Port Communication .



Chapter 3 Installation and Setup

This chapter describes the procedure for the video capture hardware and driver installation process and *ViewPoint EyeTracker*[®] software installation.

IMPORTANT: The display must be set to True Color (32 bit).

3.1 Computer System Requirements

This is the manual for *ViewPoint PC60* that runs on the Windows 2000 and Windows XP operating systems. As of version 2.8.3, we no longer officially support Windows NT, Windows 98, or Windows ME.

* DELL computers models Optiplex and Dimension are not officially supported, because historically the system BIOS in these models did not release the IRQ necessary for real-time video capture. Some newer versions of these DELL models reportedly do not suffer from this problem, however we currently have no clear specification for distinguishing between the versions. Other DELL models do not have this problem.

3.2 Video Requirements

For users who want a head free system, we recommend that the *EyeFrame*[™] hardware be purchased together with the *ViewPoint EyeTracker*[®] software for a complete solution. For users who want a head fixed system, as for psychophysics, we recommend that the *QuickClamp*[™] *EyeTracker Hardware* be purchased together with the *ViewPoint EyeTracker*[®] software for a complete solution. Either of these include a video camera and video capture device that provides 60 Hz eye movement monitoring. This is a closed system between the NTSC video camera and the NTSC video capture device so it can be used in countries that have PAL or other video standard without problem.

To work properly, the *ViewPoint EyeTracker*[®] *PC 60* version requires the special high performance PCI video capture board supplied by *Arrington Research*. It will not work with other video capture devices.

3.3 Using with Third Party Video Input Equipment

Video input may now include the PAL and SECAM standards, as well as the previously supported NTSC standard. The user should use **Video > Video Standard > *** to select the standard that corresponds to the type of video camera, videocassette recorder (VCR), etc., that is used.

The selected video standard is stored in the preferences file and will be used as the default when *ViewPoint* is next run.

The default setting is NTSC and this should not be modified unless third party video equipment is used that specifies a different video standard.

3.4 PCI Video Capture Card and Driver Installation

Important: If you are updating from version 1.x of ViewPoint EyeTracker® or have had another BT848 video capture device previously installed, you must FIRST remove all of the video capture software and drivers from your computer. SECOND, after the driver software has been removed, physically remove the old video capture device from your computer before proceeding with the new frame grabber installation.

A. Installing the New Frame Grabber

1. Turn off the computer, and then disconnect the power cable.
2. Remove the cover panel from your computer. If necessary, consult your computer system manual for instructions.
3. Remember to discharge your body's static electricity by touching the metal area of the computer chassis.
4. Select an empty PCI slot and remove the slot cover.
5. Place the card into the slot, paying particular attention that the card is inserted correctly.
6. Screw the card into place.
7. Replace the cover panel.
8. Reconnect the power cable and turn on the computer.

B. Installing the New Driver

WINDOWS XP ONLY

1. Insert the *ViewPoint EyeTracker*® CD-ROM into your CD-ROM drive.
2. If the Windows "Found New Hardware Wizard" asks you if you would like to connect to Windows Update to search for the drivers select "No, not at this time" and "Next".
3. Select "Install the software automatically" and "Next"
4. At the next dialogue box, with the top line item highlighted select "Next".
5. At the "not digitally signed" warning select "Continue Anyway".
6. Select Finish

NOTE: you will have to repeat the above steps for each input if you have a binocular or scene camera version of the eye tracker.

Other WINDOWS Operating Systems

1. If the update device driver wizard starts, click "Cancel".
2. Insert the *ViewPoint EyeTracker*® CD-ROM into your CD-ROM drive.
3. Click Start.
4. Click Run.
5. Type the following: F:\driver\lc1_2\english\disk1\setup.exe (If F is not the device letter of your CD-ROM drive, substitute with the correct drive letter).
6. Click OK.
7. Follow the onscreen instructions to complete installation.
8. Restart your computer when instructed to.

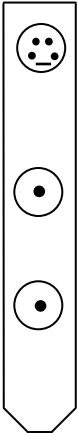
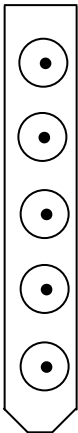
Note 1: With Windows XP it may be necessary to let the “Install New Hardware” wizard take care of the installation.

Note 2: For the binocular option and scene camera options it will be necessary to run through the driver install routine for each input. You will be prompted to do this.

Note 3: The video cable must be connected to the composite input number 1 of the frame grabber for monocular ViewPoint. Refer to Figure 4

Note 4: The video cable must be connected to the composite input numbers 1 and 2 of the frame grabber for binocular ViewPoint. Refer to Figure 4

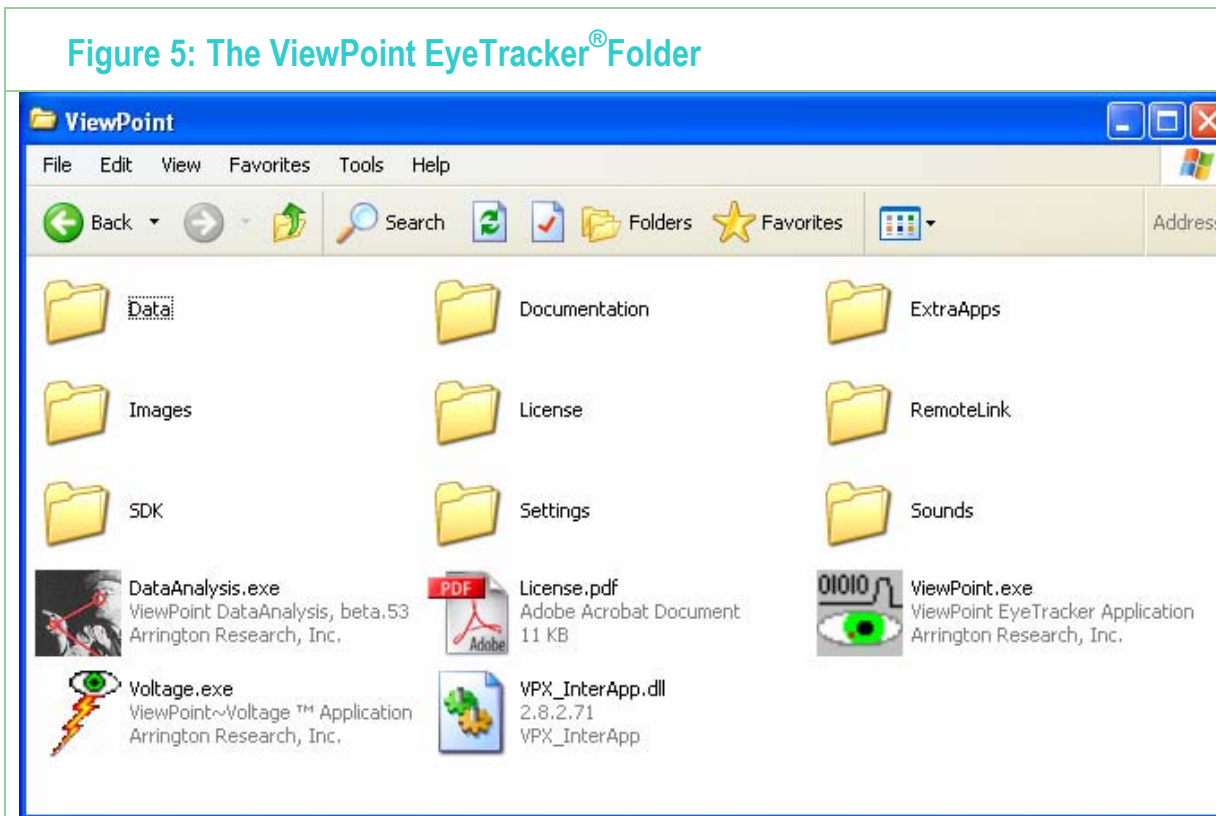
Figure 4: Frame Grabber Board Connections

Single Real-Time Input “Monocular” Card	Quad Real-Time Input “Binocular / Scene Camera Option card
 <p>Input S-VHS = Not Used</p> <p>Composite Input 2 = Not used</p> <p>Composite Input 1 = EYE_A</p>	 <p>White Sync = Not used</p> <p>Composite Input 4 = Not used</p> <p>Composite Input 3 = Scene</p> <p>Composite Input 2 = EYE_B</p> <p>Composite Input 1 = EYE_A</p>

3.5 ViewPoint EyeTracker® Software Installation

Copy the **ViewPoint** folder from the CDROM to the hard drive of your computer. This folder is illustrated in Figure 5. This directory structure must be maintained for proper functioning of the software. The *ViewPoint EyeTracker*® software will not run without the **VPX_InterApp.dll** file. Please do not make illegal copies. You may start the program immediately by double clicking the icon of the **ViewPoint.exe** application program.

Figure 5: The ViewPoint EyeTracker® Folder



3.6 License Agreement

Use of the *ViewPoint EyeTracker*® software constitutes consent to the terms of the “ARI Software License” and contained in the document **License.pdf** in the *ViewPoint* folder.

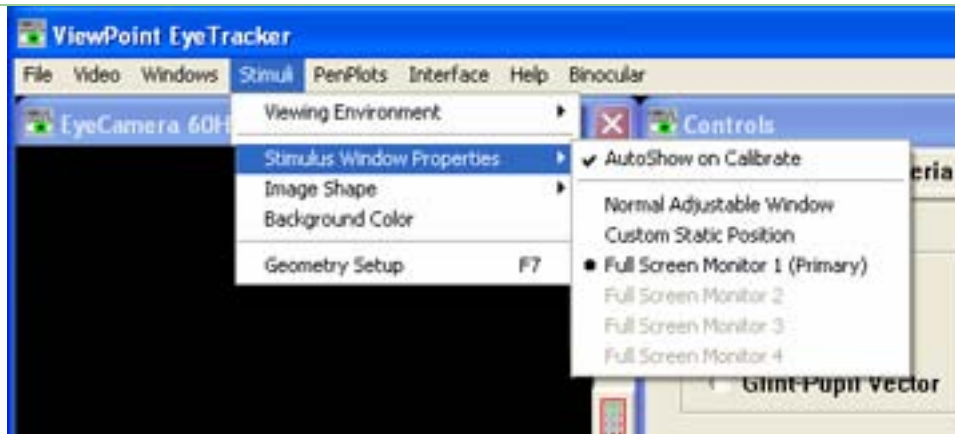
3.7 ViewPoint License (.VPL) File

The video capture board (and optional hardware) contain serial numbers that must match the one of the numbers encrypted into your ViewPoint License (**.VPL**) file. This .VPL file is located in the folder named *License* folder inside the *ViewPoint* folder. The .vpl file also specifies where any options have been enabled. The .VPL file is named in the format `YourName.vpl`.

3.8 Menu Navigation

The *ViewPoint* menu navigation system consists of various options, organized by function presented in a dropdown menu bar at the top of the *ViewPoint* application window.

Figure 6: The ViewPoint EyeTracker® Menu Navigation Bar

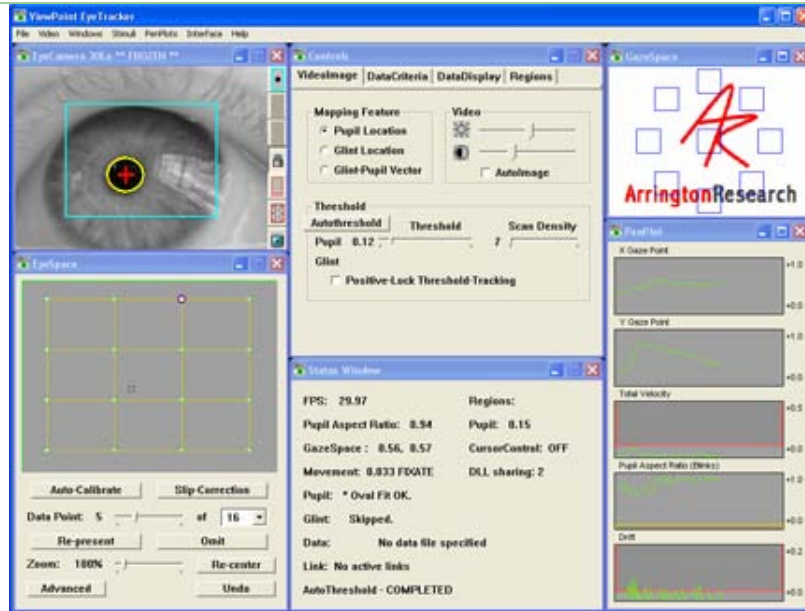


Menu Item	Use this Menu to:
File >	Open, pause and close data files, Save and load settings files, Load Images and picture lists, and Print windows
Video >	Control the incoming video signal, and Specify the tracking modes and methods
Windows >	Open and close windows.
Stimuli >	Specify the type of stimuli to be used, Control the stimulus window settings, and Specify the Geometry Grid settings
PenPlots >	Hide or show PenPlots, and Specify PenPlot window settings
Interface >	Use the cursor control feature, and Use the serial port interface (or <i>Ethernet server, coming soon</i>).
Help >	Access system configuration information, View license Holder information, Access the <i>Documentation</i> folder, and Link to the Arrington Research, Inc. web site

3.9 User Windows

When the ViewPoint EyeTracker® program is started it displays several windows arranged as shown below.

Figure 7: Start-up arrangement of the user windows



Note: The normal window layout requires a display at least 1024 x 768.

To set up for multiple monitors you will need to install a second display card into your computer and consult the operating manual for your computer for configuration settings.

Table 2: Window function descriptions	
<i>Window</i>	<i>Function:</i>
EyeCamera:	Displays the video image of the eye and image analysis graphics
EyeSpace:	Corresponds to the geometry of the EyeCamera image. This window displays an array of the relative locations of the pupil, glint, or difference vector, which were obtained during calibration. These provide information about calibration accuracy and allow rapid identification and correction of individual calibration errors by allowing manual calibration of individual points. Refer to Chapter 6.
Controls:	Allows the experimenter to adjust the image-analysis and gaze-mapping parameter settings and to specify the feedback information to be displayed in both the Stimulus window and the GazeSpace™ window. VideoImage tab : Image quality adjustments and tracking method specification. DataCriteria tab : Specify smoothing and other criteria to apply to the data. Data Display tab : Specify information to be displayed in the Stimulus and GazeSpace windows. Regions tab : Setup regions of interest (ROIs). Scene tab : Adjust brightness and contrast of scene image (scene camera option only)
Status:	Gives details about processing performance and measurements
Stimulus:	(what the subject views) that is designed to be presented full screen, preferably on a second monitor. Upon which may be displayed the subject's calculated position-of-gaze information and region of interest boxes. Refer to Chapter 8
Penplot:	Plots X and Y position of gaze, velocity, ocular torsion, pupil width, pupil aspect ratio etc. in real time. The user may select which penPlots to display using menu item PenPlot > * . The range of many of the penPlot displays can be adjusted by clicking the right mouse button in the PenPlot graph well.

Chapter 4 Tutorial

This Chapter provides a brief tutorial designed to help you get started very quickly and easily. For simplicity it assumes use of the dark pupil only method; however for normal subject testing the [Glint-Pupil Vector](#) method may be more appropriate. For further information refer to 5.1 Feature Method.

This tutorial describes the steps for using the **Head Fixed** product (which includes HMD mounted systems), not the Head Mounted product, however many of the steps are the same.

4.1 Stimulus Window Positioning (Head Fixed)

Position the monitor, on which the [Stimulus](#) window is to be displayed, so that when the subject is looking straight ahead, their position of gaze is approximately two thirds of the way up the monitor vertically and centered horizontally. The [Stimulus](#) window should be placed so that the subject can see it easily when positioned comfortably. This is best achieved using a second monitor and full screen stimulus display. Refer to [Chapter 9](#).

4.2 EyeCamera Window Setup

Instructions for proper positioning of the Camera & LED.

Position the camera at 45 degrees below the line of sight of the subject (as they are viewed from the side).

Position the LED so that it appears at the 11 o'clock position when looking at the camera lens, such that the top surface of the LED and the camera lens are along the same horizontal plane. This allows both the LED and the camera to simultaneously be as high as possible, while still not occluding the vision of the monitor.

Move the camera mount sideways, such that the LED is centered along the optical axis of the eye while the eye is looking in the center of the display. This will mean that the camera is slightly off axis (as viewed from above). In other words, if the subject looks straight down she should be looking at the LED, not at the camera lens.

Adjust the camera so that the pupil is centered in the [EyeCamera](#) window as the subject looks at the center of the display and the eyeball fills the maximum possible area as shown in Figure 7. Movement between the head and the EyeCamera must be minimized. This can be achieved by using the *Arrington Research Precision Head Positioner* and camera system.

Defocus the camera so that the corneal glint is spread to about the size of an eighth (or more) that of the pupil. Besides making the glint larger, defocusing also effectively lowers the intensity of small bright extra reflections (by virtue of the point spread function). Defocusing may be achieved by rotating the camera lens or by adjusting the slide bar to move the camera closer or farther from the eye. Generally, the image of the eye should be such that the corners of the eye (the canthi) are at the horizontal edges of the camera window.

Note that when de-focusing, it is preferable to move the focal plane farther away from the camera, rather than closer to it, because the latter will sharply focus on any dirt or debris that is on eye-glass lenses.

The LED may produce a doughnut shape of illumination. If that is the case, adjust the LED so that the darker center of the doughnut is in the center of the camera image, this will put the brighter ring around the edges near where the canthi and lids are located. This doughnut may be more easily observed by placing the palm of the hand, or a piece of paper, at the location of the eye and then moving the LED slightly.

If the video image is too dark or too bright you can adjust the contrast and brightness settings by adjusting the brightness and contrast sliders on the [Controls](#) window. When adjusting the brightness and contrast controls in *ViewPoint*, the general goal is to increase the range of gray levels as far as possible that is to DECREASE the contrast as much as possible, while MAXIMIZING the blackness of the pupil and the whiteness of the glint. However, the glint should be the only spot that is saturated to maximum brightness.

Decrease the brightness until you obtain a pupil that is as black as possible and adjust the contrast so that the glint, and only the glint, is of maximum white.

4.3 Corrective Lenses (Eye Glasses)

There are two main effects relating to the fact that the front and back surfaces of the lens will reflect light. First of all, the reflected light is wasted so that the illumination of the eye (light source path) and the image of the eye (light to camera) will be attenuated. Second the reflection from the illumination may be bounced back into the camera, which will be very bright and cause the auto-iris of the camera to produce a darker (and often varying brightness) image of the eye.

If there is a problem with the front surface reflection of corrective lenses, try adjusting the angle of the lens relative to the camera-LED assembly. There are two ways to do this:

Slightly tilt the corrective lenses by moving the earpiece so that it is near the auditory canal (hole in the ear) rather than resting on top of the ear; this is fairly easy for lightweight springy metal frames, but may not stay in place with heavier frames.

Slightly move the camera so it is more than 45 degrees below the line of sight.

4.4 Thresholding

The software attempts to locate the pupil by searching for a dark region within the pupil search area. In the [Controls](#) window : [Mapping Feature](#) group, select [Pupil Location](#). Follow the steps below to undertake the thresholding process for this method:

Toggle show threshold dots "ON" using the button in the [EyeCamera](#) window, or Menu item [Video > Show Threshold Dots](#).

Ask the subject to fixate at the center of the display screen. If self-testing move the [EyeCamera](#) window to the center of the display screen.

Select the **Pupil Search Area Adjustment** icon at the top right of the **EyeCamera** window, illustrated in [Figure 9](#): Use the mouse to drag out a rectangle that limits the area in which to search for the pupil. In particular, use this to eliminate dark shadow areas that could be confused with the pupil.

Press the **AutoThreshold** button on the **Controls** window: **threshold group**. If necessary, adjust the dark pupil threshold setting in the **Controls** window to ensure that the green dots appear only in the pupil and that the yellow oval outlines the pupil and is fairly circular. (Note that the **Status** window displays the oval's aspect ratio, 1 = perfect circle) The yellow oval indicates where the program has located the area of the pupil.

If too many dark areas other than the pupil have green dots, adjust the dark pupil threshold slider to the left. Alternatively, if insufficient pupil area is being identified as dark (with green dots), adjust the dark pupil segmentation threshold slider to the right. The **Positive-Lock Threshold-Tracking** attempts to provide optimal tracking, however it is not appropriate for all subjects or situations.

After the pupil has been isolated, adjust the scan density to use the minimum number of dots that can reliably and consistently locate the pupil. Correct thresholding of the dark pupil is illustrated in [Figure 9](#): The default set on startup is optimal for most situations and you should very rarely need to adjust this.

Note: Unnecessarily high scan density settings together with large scan areas can cause the frame rate to drop and data can be lost.

Ask the subject to look at each of the four corners of the **Stimulus** window to ensure that the pupil remains in the search box and to ensure accurate thresholding for all potential eye movements. i.e. the yellow oval outlines the pupil and is fairly circular. If self-testing, move the eye camera around to the four corners of the display to view the effects of pupil segmentation at different position of gaze. At any point the experimenter can toggle the image display **On/Off**. When **Off** it provides a still image of the eye to aid identification of successful thresholding. This is accomplished by freezing the video by (a) pressing the **Freeze Video** icon button at the bottom right of the **EyeCamera** window, (b) by selecting the menu item: **Video > Freeze Video**, or (c) by pressing the **F1** function key. When frozen, the icon button will be outlined in red and a check mark appears next to the menu item. The **EyeCamera** window will also indicate "**** FROZEN ****" in the title bar.

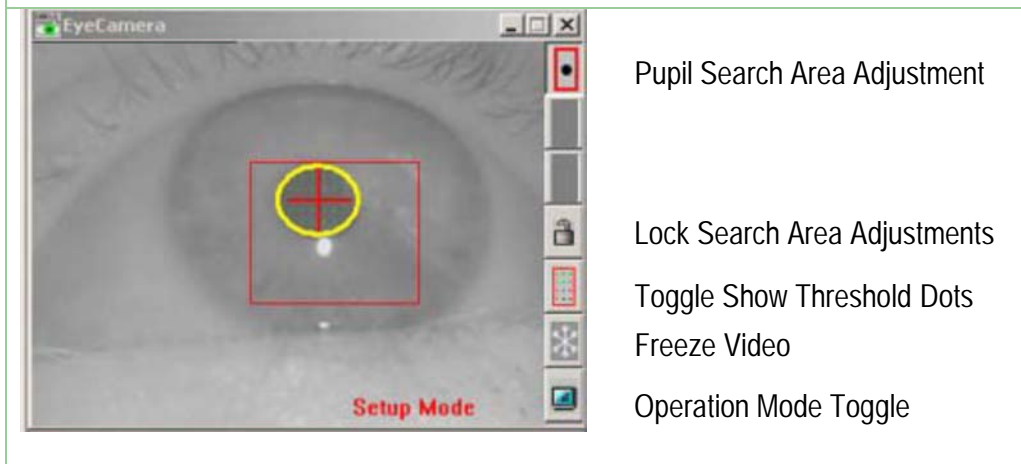
Selecting the option for **Positive-Lock threshold tracking** may help in many situations, but it does consume more CPU time.

Note: It takes time to paint the threshold dots on the screen, but it is very useful for determining what **Segmentation Threshold** and **Scan Resolution** settings are optimal. Once optimal settings have been found, the **Toggle show threshold dots** may be turned off to

remove the computational burden of painting the colored dots.

Note: this section describes pupil only thresholding. With the Pupil only method, any X, Y plane head movement will be confounded with eye movement. To measure movement that is invariant to X, Y plane head movement, use the glint-pupil vector method. For instructions on how to set up for glint-pupil vector method, please refer to Chapter 5

Figure 8: Thresholding to identify the dark pupil



4.5 Calibration (Head Fixed)

After successful thresholding as outlined in Section [4.3](#) follow the steps below:

Select menu item: Video > Mode > High Precision (30Hz 640 x 480) The EyeCamera window will display the message: "30HP".

Warn the subject of the onset of the calibration stimuli to ensure successful calibration.

Instruct the subject to look directly at the center of each stimulus until it converges to a point. The calibration stimulus points will appear in random order.

Start the calibration by pressing the [Auto-Calibrate](#) button on the [EyeSpace](#) window. The message "Get Ready" will appear briefly on the screen to draw the subject's attention to the start of the calibration process. This can be suppressed or the display time adjusted via the [Advanced](#) section in the [EyeSpace](#) window.

During the calibration process, ensure the pupil is accurately located at all times by monitoring the green dots and the yellow oval, i.e., monitoring the image segmentation.

Check the calibration by using the plot of the calibration data points in the [EyeSpace](#) window. Successful calibration will be indicated by a rectilinear and well-separated configuration of green dots corresponding to the locations of the pupil at the time of calibration point capture.

Stray calibration data points can be identified and re-calibrated. The data point slider in the [EyeSpace](#) window allows the user to select stray calibration points to be recalibrated. The active data point is highlighted in the graphics well. Data points can also be selected with the mouse by left clicking the calibration point.

Select the stray calibration point by right mouse clicking the point in the [EyeSpace](#) window.

Instruct the subject to look at the center of the stimulus and represent the calibration point by pressing the [re-present button](#) in the [EyeSpace](#) window. The message “Get Ready” will appear briefly on the screen at the calibration point location to draw the subject’s attention to the re-presentation location. This can be suppressed or the display time adjusted via the [Advanced](#) section in the [EyeSpace](#) window. This exercise can be repeated with as many calibration data points as necessary. If the calibration points are not rectilinear, for example, there are lines crossing then complete re-calibration is necessary.

A quick check of calibration accuracy may be done by asking the subject to look at particular points on the stimulus and using the [GazeSpace](#) window to verifying that the gaze point matches up with the points looked at.

4.6 Data Collection and Analysis

Now that the system is calibrated, data can be collected. To start recording data to file: Select menu item: [File > Data > New Data File](#). This will prompt you to create a new file in the directory [Data](#). When data is being stored the [Status](#) window will show: [DATA: OPEN](#) “[datafilename.txt](#)”

The menu item: [File > Data > Unique Data File](#) will create a new data file with a unique name to be opened without having to go through the File Dialog box.

Recording can be paused at any time by selecting menu item: [File > Data > Pause Data Capture](#). When data storage is paused the [Status](#) window will show: “[PAUSE](#)”. To stop recording, select menu item: [File > Data > Close Data File](#). The [Status](#) window will indicate that the file is closed.

For further information on data file formats refer to Chapter 10 [Data Collection](#)

4.7 Sampling Rate

Menu Item: [Video > Mode > *](#) can be used to select the required sampling rate. The icon button on the [EyeCamera](#) window tool bar can be used to sequence through the operating modes ([SetUp – High Precision – High Speed – Setup](#)). *Important:* Be sure to let the video mode stabilize for a few seconds, before clicking the button again. Refer also to Chapter 10 [Data Collection](#).

4.8 Frequently Used Settings

You can use the settings file [startup.txt](#) that is located in the folder named “[Settings](#)” to specify frequently used settings. For example, you may wish to specify brightness and contrast settings applicable for your subject population. When [ViewPoint](#) is launched it loads in the content of this file, which can reduce setup time.

4.9 Preferred Window Layout

A preferred startup window layout can be saved using menu item: [File > Settings > Save Window Layout](#). This can then be reloaded using the load settings menu item. Alternatively the contents of the saved settings file can be added to the startup.txt settings file so that your preferred window layout is set each time that you launch ViewPoint.

4.10 Accelerator Keys

Accelerator keys are used to make menu selections with the keyboard, rather than the mouse. The most current list can always be found within *ViewPoint* by selecting menu item: [Help > Info](#), and ShortCuts. The circumflex character '^' represents the Control key held down as a modifier key.

The user can associate an FKey with a CLP command action. This is done via the CLP interface (see section 13.23.1); for example:

```
FKey_cmd 11 dataFile_Pause
FKey_cmd 12 dataFile_Resume
```

These associations can be viewed in the Info panel: menu [Help > Info > ShortCuts](#) tab. Refer to [14.23](#)

4.11 Printing

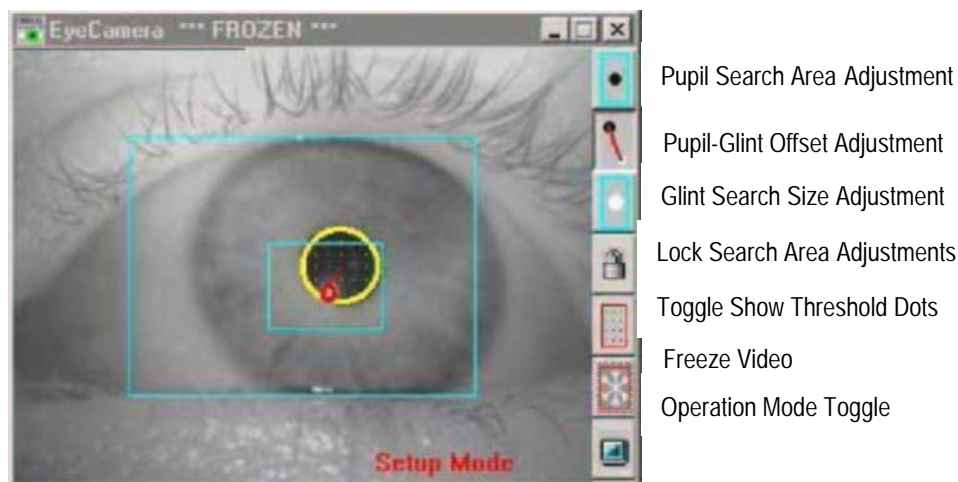
Many of the ViewPoint windows can be printed using menu item: [File > Print > ...](#) To include the current date and time on the prints, check menu item [File > Print > DateTimeStamp Printouts...](#)

Note: you may want to select Freeze before Print to prevent pull down menu occlusion.

Chapter 5 Locating the Pupil and Glint

The *EyeCamera* window displays the video-image of the eye, as well as overlay-graphics that graphically provide information about image segmentation and performance. This is shown in Figure 9: The overlay graphics include: thresholding results (e.g., green dots indicating dark areas), pupil location and diameter calculation results (yellow oval fit to pupil), corneal reflection location results. The mouse is used in this window to pull (drag-out) a rectangle to define a limited search area, sometimes called a *gate*, for both the pupil and glint (see 5.5 Setting the Search Regions)

Figure 9: EyeCamera Window



Search regions are defined by dragging out a rectangle with the mouse.

5.1 Feature Method

A *feature* is any content in the video image that can be identified, located and tracked; these include the dark pupil and the bright corneal reflection, i.e., glint. The *Feature Method* combo box allows the experimenter to select which features to use. This combo box is on the *Video Image* tab of the *Controls* window. The feature methods fall into two basic categories: single data point methods and multiple data point methods. Which method to use is probably best determined through experimentation.

The single data point methods, for example using *Pupil Location* only or specular corneal *Glint Location* only are sensitive to slight sideways head movements. The *Glint Location* method is normally useless, but is included for educational purposes.

The multiple data point method uses the *Glint-Pupil Vector* Difference between (i) the center

of the pupil and (ii) the center of the specular corneal reflection. This is more robust against small movements of the subject within the tracking apparatus. The corneal reflection (CR) and the dark pupil (DP) move together as the head moves (translates in the x-y plane that is normal to the optical axis of the camera). By taking the vector difference between these two signals, a relatively translation-invariant point-of-regard eye tracking system can be achieved.

The disadvantages of the [Glint-Pupil Vector](#) method are (a) there are now two sources of video and segmentation noise instead of one, (b) given a change in viewing direction, the vector variation is smaller than the variation of the pupil (or the glint) alone. The result is a lower signal to noise ratio. Moreover, the vector method is sensitive to a different type of translation error. The [Glint Location](#) and [Pupil Location](#) methods are particularly sensitive to translation of the head in a horizontal (sideways, x-axis) or vertical (up/down, y-axis) direction and less sensitive to in-and-out (closer or farther from the camera, z-axis) movement of the head. By contrast, the [Glint-Pupil Vector](#) method is robust against x-axis or y-axis movement, but is more sensitive to z-axis movement of the head, because this affects the length of the calculated vector; that is, the vector becomes shorter as the head is moved backward away from the camera. This is particularly true when the camera is close to the eye, because the angular field of view is wider. If a more remote camera is used it will be less sensitive to Z-axis variation.

The [Feature Method](#) combo box also includes a [Slip Compensation](#) method described in section 5.3, and data Simulation options described in section 5.4.

5.2 Noise

The major problem that the user will face is to increase the signal to noise ratio. In the [Glint-Pupil Vector](#) mode, the basic measure of signal is the length of the vector from the center of the pupil to the center of the glint. Noise comes from many things, for example: eyelid droop, eye blinks, extreme direction of gaze angles that produce a reflection from the less smooth sclera (white part of the eye), shadows, extraneous specular reflections, as well as the internal electrical video noise.

The first most obvious way to increase the signal is to move the camera closer or zoom in, so the pupil to corneal reflection difference vector appears larger in the [EyeCamera](#) window. The trade-off here is that smaller head movements can move the eye out of the view of the camera. The mapping function is calculated based on the calibration data, and it is only as good as the calibration. There are many sources of non-linearity. Consequently, mapping the raw eye-image-feature data to direction-of-gaze requires a sufficiently sophisticated non-linear mapping function. Obviously, if the subject is not looking at the calibration point when the data is sampled, then the calibration function that is calculated is not going to produce accurate direction of gaze information.

5.3 Automatic Slip Compensation

This method combines the advantages of the larger signal space and reduced signal noise of the [Pupil Location](#) method, together with the translation-error robustness of the [Glint-Pupil Vector](#) method. Select [SlipCompensation](#) from the [Feature Method](#) pull down menu on the [Controls](#)

window [VideoImage](#) tab. *Note: do not confuse this with calibration Slip Correction.*

5.4 Simulation of Gaze

5.4.1 Manual Simulation

The user can use the mouse to simulate position of gaze. Select [Manual Simulation](#) from the [Feature Method](#) pull down menu on the [Controls](#) window [VideoImage](#) tab. Click and hold the mouse button in the [GazeSpace](#) window. Note that the smoothing operation is still applied, so unless smoothing is set to one (turned off) it may take several clicks at a location before the gaze point indicator moves to the location of the mouse. The pupil and glint quality codes are set to best-quality (as soon as the user moves the mouse in the [GazeSpace](#) window) so that subsequent operations are not impeded. *Note: the eye video must be frozen.*

5.4.2 Pattern Simulation (*only on special versions*)

This provides simulated eye positions data in the form of a continuous test pattern that is useful for developing and debugging interfaces, such as layered SDK program interfaces, serial port transfers, Ethernet client / server connections, etc. (*Note this is designed to replace the option: Interface > Serial Port > Send Test Pattern, with a more general, earlier stage, simulator.*)

5.5 Setting the Search Regions

The eye tracker's performance is significantly increased as the size of the search areas is reduced. Limiting a search area also provides a convenient way to eliminate areas with video noise, extraneous specular reflections or dark shadow areas, any of which may confuse the location process. Buttons on the side of the [EyeCamera](#) window are used to select the adjustment mode for:

- pupil search area,
- glint search size, or
- pupil-glint vector offset.

These top buttons act like "radio buttons", so only one selection may be made at a time. Depending on which mapping mode is selected, the user can easily adjust any of these by simply using the mouse to drag out the rectangle or vector within the [EyeCamera](#) window, as illustrated in Figure 8: Thresholding to identify the dark pupil

You can lock the adjustments to protect them from accidental changes by selecting the "Lock all adjustments" icon button.

When the mapping feature mode is set to [Pupil Location](#), the pupil is found by searching for the dark region within the pupil search area. To adjust this area, first press the button to select pupil search area adjustment mode, then use the mouse in the [EyeCamera](#) window to drag out the smallest rectangle that catches the pupil over its full range of movement.

When the mapping feature mode is set to [Glint-Pupil Vector](#) the large pupil must be found

first as above, then the smaller corneal reflection is searched for relative to the pupil:

Make *pupil scan area* and *pupil threshold adjustments* to get a good lock on the pupil boundaries. Make sure that the software keeps a good lock on the pupil as the subject looks at the four corners of the screen.

Press the button to select pupil-glint offset vector adjustment mode, then use the mouse in the [EyeCamera](#) window to drag out the offset vector: drag a line from the center of the pupil to the center of the corneal glint.

Press the button to select *glint search size adjustment* mode, then use the mouse in the [EyeCamera](#) window to drag out smallest rectangle that catches the glint at all possible eye positions. Because the glint search size moves relative to the calculated center of the pupil, the absolute placement of the glint search size specification rectangle with the mouse is not important, only its size is important.

The oval fit algorithm will continue to fit the pupil beyond the limits of the pupil search area. Consequently, the pupil search area can be substantially smaller than may initially be imagined.

5.6 Brightness and Contrast Adjustments

The default values of the video image Brightness and Contrast should provide good operation for most subjects when used with the close focus camera system provided by Arrington Research. However, with other camera configurations you may need to make fine adjustments. Also, subjects with very dark iris pigmentation may need the brightness reduced slightly. Adjust to provide optimal segmentation of the pupil and iris with gradually varying gray scale. High contrast is not as good as a large range of gray scale values.

5.6.1 Video AutoImage

The [AutoImage](#) check box located on the [Controls](#) Window [Video Image](#) tab. When checked *ViewPoint* will automatically adjust the brightness and contrast values to optimal settings. Only the region within the pupil scan area is examined, so the pupil scan area rectangle must be of sufficient size for the algorithm to sample a range of gray levels, otherwise the algorithm will fail.

5.7 Thresholding and Setting the Scan Density

Thresholding

The program scans over the video image for the dark pupil and / or for the light corneal reflection. Adjusting the Threshold sliders on the [Controls](#) window: [Threshold group](#) controls which luminance values to include or exclude. The pupil threshold slider adjusts the threshold level (sensitivity) for the pupil. Moving the slider to the right raises the dark pupil threshold ceiling, allowing more (lighter) gray levels to be counted as part of the dark pupil. The glint threshold slider adjusts the threshold level (sensitivity) for the white corneal reflection. Moving the slider to the left lowers the light reflection threshold floor, allowing more (darker) gray levels to be counted as part of the reflection.

Clear focus is not always optimal for obtaining good segmentation, because display screen reflections over the pupil can sometimes confuse the image segmentation process. Also, defocusing can cause the specular corneal glint to appear larger, which can make it easier to locate.

5.7.1 Setting the Scan Density

The resolution at which the program samples pixels in the video image is adjusted by the Scan Density sliders. The finest resolution is with the slider set to the left, as the slider is moved to the right, only every n'th pixel is examined, where n is the number of clicks to the right. The *ViewPoint EyeTracker*® works by isolating the pupil and corneal reflection in the video image. To segment the image properly, the intensity-threshold levels must be appropriately set. The pupil and the corneal reflection are located by first taking the mean position of all sample points within threshold limits. The spatial resolution of the sampling may need to be adjusted to optimize speed or accuracy. Moving the slider to the right increases the sample spacing (coarse) which reduces the sampling resolution and correspondingly the number of dots shown. Moving the slider to the left increases the resolution (fine). The result of the sampling resolution is graphically displayed when the Show Threshold Dots button on the *EyeCamera* Window or Menu item **Video > Show Threshold Dots** is selected.

Note: It takes time to scan pixel values and to paint them. It is very useful for determining what Segmentation Threshold and Scan Resolution settings are optimal. Once optimal settings have been found, the scan density should be reduced as far as possible to reduce the computational burden. If fine scan resolution and a large scan area is required, then the Show Threshold Dots may be turned off to remove the computational burden of painting the colored dots.

5.7.2 AutoThreshold

Pressing the **AutoThreshold** button automatically sets desirable Light-Reflection and Dark Pupil threshold levels. After this is done, the user may want to make further adjustments, which is easily done using the sliders.

5.7.3 Positive Lock Threshold Tracking

Positive Lock provides continuous automatic feature threshold adjustment. To activate this feature select **Positive-Lock Threshold-Tracking** check box on the **Controls** Window **Video Image** tab. Note: this only adjusts the pupil threshold; the glint threshold must be adjusted manually.

5.8 Criteria

ViewPoint provides several criteria for accepting or rejecting data. Using these wisely and judiciously can substantially improve the performance in many situations. These can help tag messy blink data so that it can be easily eliminated from post-hoc analysis. They can also improve real-time performance and help protect equipment; for example if a galvanometer is connected to the eye positions signals. The following sections discuss various criteria available.

The circle fit around the pupil in the [EyeCamera](#) window changes from yellow to orange when the specified criterion is violated.

5.8.1 Pupil Aspect Criterion

The program can reject a dark segmented area as being the pupil based on its ability to fit a circle to that area. If the ratio of the minor-axis to the major-axis is less than the criterion level, then that area is rejected. If you are going to use this feature, then typically a criteria level of 0.6 is a good place to start. Enable the [Pupil Aspect Ratio](#) time-plot in the [PenPlot](#) window to graphically view the criterion level relative to the pupil aspect ratio data-value in real-time. Adjust the [Pupil Aspect Criterion](#) slider so that the threshold bar is below the data value for all potential eye movements, but above that for blinks. The pupil oval fit changes color from yellow to orange when criterion is violated.

The variation of aspect ratio over the range of eye movements will depend on the viewing angle of the camera. The eye image in [Figure 9](#): was taken with a micro camera (arranged as in [Figure 1](#): Schematic of the *ViewPoint EyeTracker*[®] System). The pupil will appear more oval as the angle increases between the optical axis of the camera and the line of sight of the eye.

5.8.2 Width Criteria

ViewPoint also provides [Minimum Pupil Width](#) and [Maximum Pupil Width](#) criteria. Enable the [Pupil Width](#) time-plot in the [PenPlot](#) window to graphically view these criteria levels relative to the pupil width data-value in real-time. Adjust the [Maximum Pupil Width](#) and the [Minimum Pupil Width](#) sliders so that the data value is between the threshold bars for all potential eye movements.

5.9 Step-by-step guide for *Glint-Pupil Vector* method

This section assumes that you have successfully completed the Chapter 4 tutorial on locating the pupil.

In the Controls window select Glint-Pupil Vector.

Ask the subject to fixate at the center of the display screen. If self testing, move the [EyeCamera](#) Window to the center of the display screen.

Select the [Pupil Search Area Adjustment](#) icon at the top right of the [EyeCamera](#) Window, illustrated in [Figure 9](#):

Use the mouse to drag out a rectangle that limits the area in which to search for the pupil. In particular, use this to eliminate dark shadow areas that could be confused with the pupil. If necessary adjust the brightness and contrast settings and move the illuminator to provide uniform illumination over the eye image.

Press the [AutoThreshold](#) button on the [Controls](#) window. If necessary, adjust the dark pupil threshold setting in the Controls window to ensure that the green dots appear only in the pupil and that the yellow oval outlines the pupil and is fairly circular. (Note that the [Status](#) Window displays the oval's aspect ratio, 1 = perfect circle) The yellow oval indicates where the program has located the area of the pupil.

If too many dark areas other than the pupil have green dots, adjust the dark pupil threshold slider to the left. Alternatively, if insufficient pupil area is being identified as dark (with green dots), adjust the dark pupil segmentation threshold slider to the right.

After the pupil has been isolated, adjust the scan density to use the minimum number of dots that can reliably and consistently locate the pupil. Correct thresholding of the dark pupil is illustrated in [Figure 8](#):

Ask the subject to look at each of the four corners of the [Stimulus](#) window to ensure that the pupil remains in the search box and to ensure accurate thresholding for all potential eye movements. i.e. the yellow oval outlines the pupil and is fairly circular. If self testing move the eye camera around to the four corners of the display to view the effects of pupil segmentation at different position of gaze.

Select the Pupil-Glint Offset Adjustment icon on the [EyeCamera](#) window, illustrated in [Figure 9](#): Use the mouse to drag out the offset vector from the center of the pupil to the center of the corneal glint.

Press the button to select the glint search size adjustment mode, and then use the mouse in the [EyeCamera](#) window to drag out smallest rectangle that catches the glint at all possible eye positions. Because the glint search size moves relative to the calculated center of the pupil, the absolute placement of the glint search size specification rectangle with the mouse is not important, only its size is important.

After the glint has been isolated, adjust the scan density to use the minimum number of dots that can reliably and consistently locate the glint. Correct thresholding of the glint is illustrated in [Figure 9](#):

Ask the subject to look at each of the four corners of the [Stimulus](#) window to ensure that the glint and pupil both remain in the respective search boxes and to ensure accurate thresholding for all potential eye movements. i.e. the yellow oval outlines the pupil and is fairly circular and the red oval outline the glint and is fairly circular. If self testing move the eye camera around to the four corners of the display to view the effects of pupil and glint segmentation at different position of gaze.

At any point the experimenter can toggle the image display [On/Off](#). When [Off](#) it provides a still image of the eye to aid identification of successful thresholding. This is accomplished by either (a) pressing the [Freeze Video](#) icon button at the bottom right of the [EyeCamera](#) window, or (b) by selecting the menu item: [Video > Freeze Video](#). When frozen, the icon button will be outlined in red and a check mark appears next to the menu item. The [EyeCamera](#) window will also indicate ***** FROZEN ***** in the title bar.

Scan adjustments can be locked by selecting the icon on the [EyeCamera](#) window tool bar.

5.10 Alternative Segmentation Methods

Various algorithms are available for determining the pupil center. Lighting conditions and performance considerations will determine which method is best for a particular job. The menu

item: [Video > Pupil Segmentation Method > ***](#) is used to make a selection.

5.10.1 Centroid

Centroid means “center of mass”. This is a simple method that may be useful if there is difficulty discriminating the edge of the pupil, or if the pupil is very small. To use this method select the

The pupil location is at the average position of all points above threshold, weighted according to how much above threshold. Points below threshold are weighted more if they are darker. This centroid location is used as the starting point for its additional processing by the more sophisticated methods discussed in the next sections. .

5.10.2 Edge Trace (*only on special versions*)

The pupil location is the center of the extreme values obtained during an edge trace around the pupil, starting from the point at 3 o'clock from the weighted centroid. This algorithm first scans rightward to find the right edge of the pupil. Next the algorithm traces the edge of the pupil, using the dark pupil threshold limit as the edge criterion. The extreme positions obtained during the edge trace are used to fit an oval, the center of which is taken to be the pupil location.

5.10.3 Oval Fit

This method scans the area around the centroid for extreme left, right, top and bottom values that are used as the coordinates of an unrotated (flat) bounding rectangle. The center of this bounding rectangle is taken as the center of the pupil. This method is more robust than the centroid method alone, and it takes less cpu time than the general rotated Ellipse method discussed next.

5.10.4 Ellipse (*only on special versions*)

This provides a general rotated ellipse, so that a good fit is made to oblique pupil images. It requires significantly more cpu time than the Oval Fit method, but in some situations it can provide a more accurate calculation for the pupil center. Because there are more degrees of freedom, the ellipse may appear more wobbly or jelly-like than with the Oval Fit method, however the pupil center calculations is usually more accurate, because more points are used for fitting.

5.10.5 Glint Segmentation Methods

The previous sections have focused segmentation and identification methods for the pupil. It is also possible to change the default segmentation method for the glint, though it is not normally required. This is done with menu item: [Video > Glint Segmentation Method > ***](#) .

5.11 Pupil Scan Area Shape Options

The user can specify whether to change the scan area for the pupil to either rectangular or elliptical using CLP commands. Elliptical scan area can be effective at eliminating dark spots that the software may interpret as a pupil. Refer to [14.6](#).

Chapter 6 Calibration

Before calibration, the pupil and corneal reflection must have been isolated with appropriate threshold settings.

6.1 Raw Data Without Calibration

The *ViewPoint EyeTracker*[®] starts up in a **coarsely calibrated** state that provides precise timing of raw (uncalibrated) eye movements. This is sufficient for many applications that can utilize relative eye movements, such as quadrant-wise “preference of looking” tasks. If your application requires more precise *gaze point* information, then further calibration will be required.

6.2 Calibration Description.

Raw pupil and corneal reflection locations do not indicate where the subject’s position-of-gaze is. These raw data points in **EyeSpace**[™] (i.e. the *video-display space*) must be mathematically mapped to the subject’s **GazeSpace**[™] (i.e. the *visual-stimulus space*).

When using the **Glint-Pupil Vector** method it is still important to obtain separate calibrations for each individual, because of individual variations in corneal curvature.

Calibration stimuli are presented to the subject in the **Stimulus** window and also indicated to the user in the **GazeSpace** window. The subject should be instructed to foveate each point in turn, so the system can determine appropriate coefficients for the mathematical mapping. The calibration mode is “Tunnel Motion” calibration, where shrinking motion of a rectangular frame captures the subject’s visual attention and smooth pursuit brings the subject’s gaze point to each of the desired calibration spots in turn.

6.3 Calibration Carryover.

There substantial similarities between the eyes of different people, so it is sometimes possible to calibrate the system to one person, who is easy to calibrate, and then obtain reasonable data using that calibration for another person – though you will probably at least want to do a Slip Correction (see § 6.8).

When using the **Glint-Pupil Vector** method you may still want to obtain separate calibrations for each individual, because of individual variations in corneal curvature.

6.4 Calibration Procedure (Head Fixed)

For a step by step guide to successful calibration refer to [4.5. Calibration](#)

Menu Item: **Video > Mode > High Precision (30 Hz 640 x 480)** should be selected prior to starting calibration to ensure the highest degree of accuracy. The icon button on the **EyeCamera**

window tool bar can be used to sequence through the operating modes ([SetUp – High Precision – High Speed – SetUp](#)).

The [EyeCamera](#) window can be completely hidden or minimized at any time during the eye tracking process without disrupting video capture.

Calibration is started by pressing the [Auto-Calibrate](#) button in the [EyeSpace](#) window. If menu item: [Stimuli > Stimulus Window Properties > AutoShow on Calibrate](#) is also selected, then the [Stimulus](#) window will automatically be displayed full screen on the primary monitor. The option should be turned Off when assigning the [Stimulus](#) window on a secondary monitor.

The message “Get Ready” will appear briefly on the screen to draw the subject’s attention to the start of the calibration process. This can be suppressed or the display time adjusted via the [Advanced](#) section in the [EyeSpace](#) window.

The automatic calibration sequence may be stopped by pressing the [STOP Calibration](#) button in the [EyeSpace](#) window. Pressing the ESC key will both stop the calibration and remove the full screen display if it is on the primary monitor.

The number of calibration points is selected by the user using the pull-down menu item in the [EyeSpace](#) window. The number of calibration points may be set to: 6, 9, 12, 16, 20, 25, 30, 36, 42, 49, 56, 64 or 72. A higher number of points may help with subjects that have corneal abnormalities or difficulty foveating. The current setting is indicated by a check mark. A setting of 12 or 16 is usually quite adequate. With fewer calibration points, good calibration accuracy is essential for each point. When a large number of points are used, the effect of any single point will be less. In general try to use at least 9 points to obtain a good calibration.

Speed of presentation of the calibration stimulus points can be adjusted via the [Advanced](#) section in the [EyeSpace](#) window.

6.5 Assessing Calibration Success

A quick check of calibration accuracy may be done by asking the subject to look at particular points on the stimulus and using the [GazeSpace](#) window to verify that the gaze point matches up with the points looked at.

The arrangement of calibration data points in the [EyeSpace](#) window provides a method of assessing how good the calibration data is.

Successful calibration will be indicated by a relatively rectilinear and well separated configuration of dots. The mapping method (selected in the [Controls](#) Window) determines how these data points are plotted. If [Pupil Location](#) is selected, then the plot shows green dots corresponding to the locations of the pupil at the time of calibration point capture. The dots are joined by yellow lines that indicate the spatial relationship between the dots. If [Glint-Pupil Vector](#) is selected, the plot shows blue dots corresponding to the locations of the pupil at the time of calibration point capture, but now they are shifted so that they are all relative to the corneal glint (red dot) that is plotted at the center of the data point chart. The dots are joined by yellow lines that indicate the calibration index number (which is the order in which they are presented if sequential

presentation order is selected).

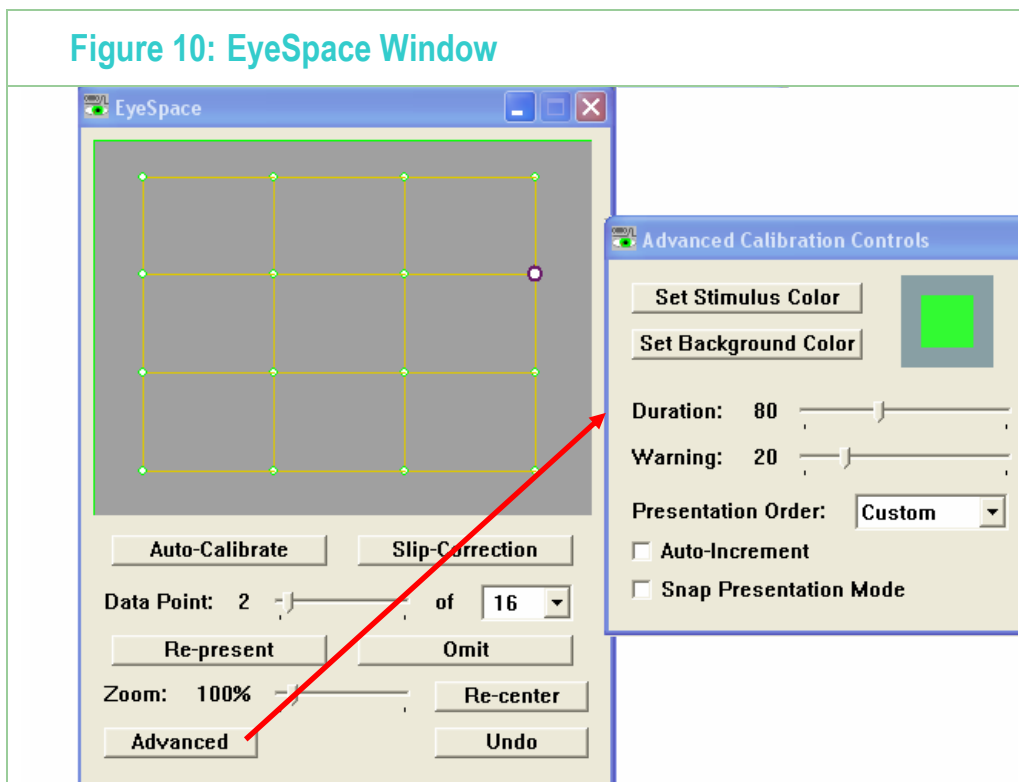
The **EyeSpace** window is shown in Figure 10:.

The top part of the window shows a 320x240 graphics well that represents of the coordinate space of the **EyeCamera** window. If **Pupil Location** or **Glint Location** modes are selected, a black square shows the current pupil location, or the current glint location, respectively. In **Glint-Pupil Vector** mode the black square shows the vector difference between the pupil and glint locations, with the glint end of the vector fixed at the red dot.

For ease of data point viewing, the calibration data points and real-time feature points (pupil and / or glint) can be zoomed in or out using the **Zoom** slider, and can easily be moved by dragging with the *right mouse* button. The **Re-center** button repositions the data points in the center of the graphics well.

6.6 Omitting Individual Calibration Points

The **Omit** button in the **EyeSpace** window will allow the user to omit an individual calibration data point from the mapping calculations. This may be required if for some reason an individual calibration point is far out of the rectilinear distribution and re-present is not successful. Use the data point slider or mouse click to select the required data point. Pressing the **Restore** button restores the omitted point.



6.7 Re-presenting Individual Calibration Data Points

The data point slider allows the user to select individual (e.g., stray) calibration points to be recalibrated. The active data point is highlighted in the graphics well. Data points can also be selected by left clicking the mouse. To re-present the selected (stray) calibration point, press the **Re-present** button in the **GazeSpace** window.

The message “Get Ready” will appear briefly on the screen to draw the subject’s attention to the location of the calibration point. This can be suppressed or the display time lengthened via the Advanced section in the EyeSpace window or by using settings files. Refer to Chapter 12.6.

6.8 Slip Correction

During data collection, the subject may move in the X and Y planes such that the measured position of gaze no longer corresponds to actual position of gaze. This type of problem can usually be corrected easily by translating (shifting) the calibration data set. First be sure to select (click with the mouse) a good calibration point near the center of the display. The **Slip-Correction** button in the **EyeSpace** window will re-present the currently selected calibration point to the subject and automatically adjust the remaining points to compensate for the measured slip in the (x,y) plane.

The message “Get Ready” will appear briefly on the screen to draw the subject’s attention to the location of the calibration point. This can be suppressed or the display time lengthened via the Advanced section in the EyeSpace window or by using settings files. Refer to Chapter 12.6.

Slip-Correction is generally not required when using the **pupil-glint vector difference** method; it is most useful when using the pupil-only or glint-only methods.

Note that **Slip-Correction** is a different from **Slip-Compensation** feature method described in section 5.3.

6.9 Instructions to Subject

For auto-calibration, it is usually preferable to randomize the presentation order of the calibration points, which is the default setting. With **Sequential Presentation Order** of calibration stimulus points, a leading source of calibration error is that the subject anticipates the presentation location of the next point, before the current stimulus point has finished. Explain to the subject that it is important to fixate on the calibration stimulus point until the point has completely disappeared.

The calibration stimulus presentation order type may be changed. Press the Advanced button in the EyeSpace window for access to the controls. See section 6.12.1.

6.10 Dominant Eye

If the subject is known to have a dominant eye, a better calibration is obtained if this eye is used.

6.11 Saving Calibration Eye Images

Troubleshooting calibration difficulties can be made easier by viewing the eye image at the time the calibration data point is taken. The following CLP command saves these eye images in a folder named "*Calibration*":

```
SaveCalibrationEyeImages Yes
```

This command will also create the folder if it does not exist.

6.12 Advanced Calibration Controls

The user can change the default color settings of the calibration stimulus rectangles and the background using the [Advanced Calibration Controls](#) window. This window is reached from the [Advanced](#) button on the [EyeSpace](#) Window.

The [Duration](#) slider specifies the approximate duration in milliseconds of each of the concentric contracting "tunnel motion" calibration stimulus rectangles.

6.12.1 Presentation Order

The [Presentation Order](#) pull-down menu allows the user to choose to present the calibration stimulus points in one of three orders. The default presentation mode is Random.

Sequential: Calibration stimulus rectangles are presented from the top left hand corner of the screen to the bottom right hand corner of the screen.

Random: Calibration stimulus rectangles are presented in random order. The series is re-randomized every time the set finishes, so that there is a new random order for the next loop.

Custom: Using CLP commands the user can specify the presentation order of the calibration stimulus rectangles. See section 14.8.14 to 14.8.16.

6.12.2 Snap and Increment Calibration Modes

If manual calibration is preferred, then the user can select [Snap Presentation](#) mode. In this mode the currently selected calibration data point is active and the [Re-present](#) button will immediately perform the calibration based on the eye position at the time. There is no warning and no calibration stimulus point presented.

When in this mode the behavior of the [Re-Present](#) and the [Slip-Correction](#) buttons/commands is changed as described and is indicated by the appearance of an asterisk (*) on these buttons.

When operating in this mode the user can choose whether to automatically advance to the next calibration data point by selecting the [Auto-Increment](#) button. This mode is indicated by the appearance of *++ on the [Re-Present](#) and the [Slip-Correction](#) buttons. If [Auto-Increment](#) is not selected then the selected calibration point will not change. The data point advanced to is determined by the [Presentation Order](#) mode selected.

6.12.3 Adjusting the Calibration Area

The user may want to adjust the size and position of the area within which the calibration stimulus points are presented. This is especially useful when part of the display screen is occluded, as in fMRI environments. In the [Advanced Calibration Controls](#) Window press the [Adjust Calibration Area](#) button. Refer to Figure 10:EyeSpace Window. This opens the Regions Tab on the [Controls](#) window. Use the left mouse button in the [GazeSpace](#) window to drag out the required size and position of the calibration area. The size and position coordinates are displayed in the [Regions](#) tab and in the [GazeSpace](#) window. Refer to section 14.8.26 for details of command `calibrationRealRect`. The [Revert](#) button will undo the last change and the [Default](#) button will return the calibration areas size to the default setting.

6.13 Custom Calibration Point Positions

ViewPoint now allows the user to specify the locations of the calibration stimulus points. This can be very useful if you need to avoid certain visual obstacles. Loading in the custom calibration stimulus point locations does not automatically switch the program to use these points. This is done with a separate command `calibration_CustomPointsUsed True`, see section 14.8.22.

The nearest-neighbor grid-lines in the EyeSpace are only useful if the pattern of stimulus points is in a rectilinear grid. The nearest-neighbor grid-lines are not automatically drawn when this option is used, because the points could be in any configuration. The drawing of these lines can be toggled on/off with CLP commands. See section 14.8.24.

The calibration point index is in column-major order, which means that the points run down the columns before go to the next row. For example:

1	4	7	10
2	5	8	11
3	6	9	12

6.14 Geometry Grid

The Geometry Grid is used for calculating the eye movements in degrees (rather than screen coordinates), and for helping the user visualization of the size of the stimuli in the display.

First, open the [Stimulus](#) window full screen on the selected monitor. Then select menu item: [Stimuli > Geometry Setup](#) to display the [Apparatus Geometry](#) window and to show the geometry grid lines on the [Stimulus](#) window.

The grid lines are presented as light blue lines separated by one degree of visual arc. The grid spacing depends on the viewing distance and also on the horizontal and vertical size of the image of the [Stimulus](#) window on the monitor. *ViewPoint* provides an easy way to take these measurements and to enter them, so that the program can perform the trigonometry and display the grid lines accurately.

The Stimulus window should be made full screen size on the display that will be used for

visual stimulus presentation. If this has not been done, the Stimulus window will be set to full screen on the primary monitor when the **Apparatus Geometry** window is raised. When the **Apparatus Geometry** window (see [Figure 11:](#)) is displayed, the **Stimulus** window will display two thick red lines, one vertical and one horizontal. The **Apparatus Geometry** window contains three sliders. The user should adjust these sliders to indicate:

- The viewing distance from the subject's eye to the center of the display screen,
- The lengths of the horizontal and vertical red lines in the **Stimulus** window.

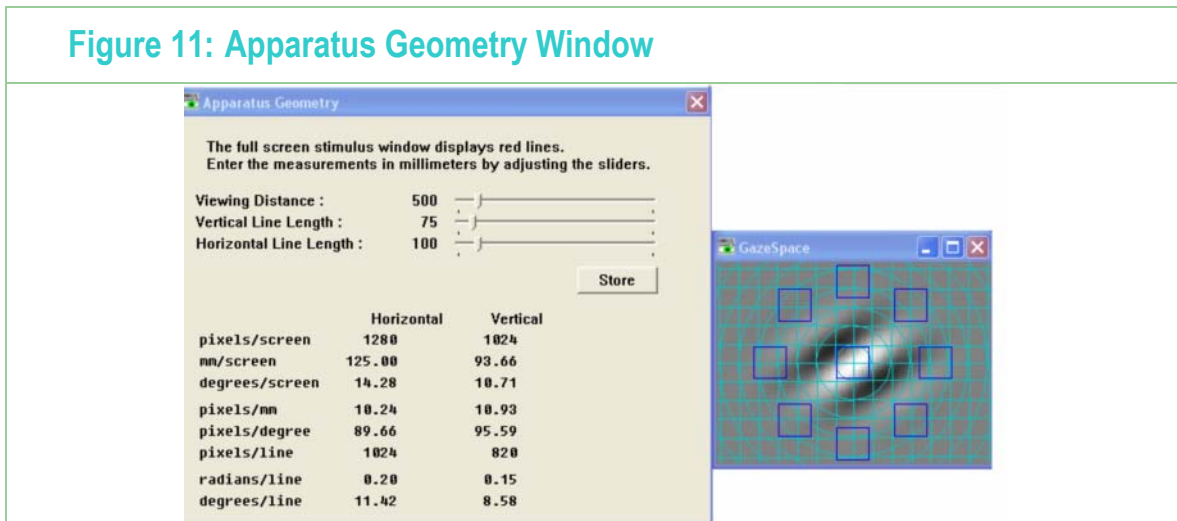
The units of measurement are arbitrary, but they must be consistent, e.g. all measurements in millimeters, or all measurements in inches. The **Apparatus Geometry** window also displays various numerical calculations that may be useful.

After adjustments have been made, the measurements should be saved by pressing the **Store** button. Subsequent runs of *ViewPoint* will maintain the stored settings.

Note that the one-degree lines are accurate on the **Stimulus** window, however the “one-degree” lines on the **GazeSpace** window are scaled and may show a miniature view of the **Stimulus** window.

The geometry gridlines can be displayed on the **Stimulus** window and on the **GazeSpace** window without the **Apparatus Geometry** window being active, by using the check boxes on the **Controls** window **DataDisplay** tab.

Figure 11: Apparatus Geometry Window



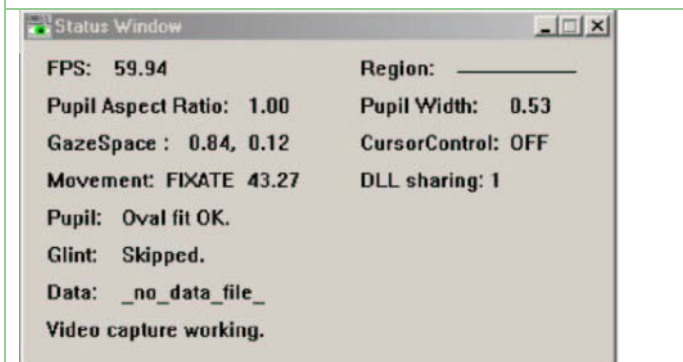
Chapter 7 Cursor Control

The **CursorControl** feature will allow the subject to move the cursor with the position of gaze of their eyes. It only works with head fixed hardware (e.g., the *QuickClamp*) or with head mounted display (HMD) hardware.

Select menu item: **Interface > CursorControl > Eye Moves Mouse**. The **Status** window will display **CursorControl: ON**. Menu item **Interface > CursorControl > Fixation Clicks Buttons** when toggled ON will cause a button click event to be issued when the fixation duration reaches a pre-set dwell time. The “dwell time” in seconds can be specified using the **MouseClicked If Fixated** slider in the **Data Criteria** panel of the **Controls** window and also specified using settings files. Menu item **Interface > CursorControl > Blinks Clicks Buttons** when toggled ON will cause a button click event to be issued when the eye tracker detects a blink.

Note: Increasing the amount of smoothing will substantially increase the usability of the cursor.

Figure 12: Status Window



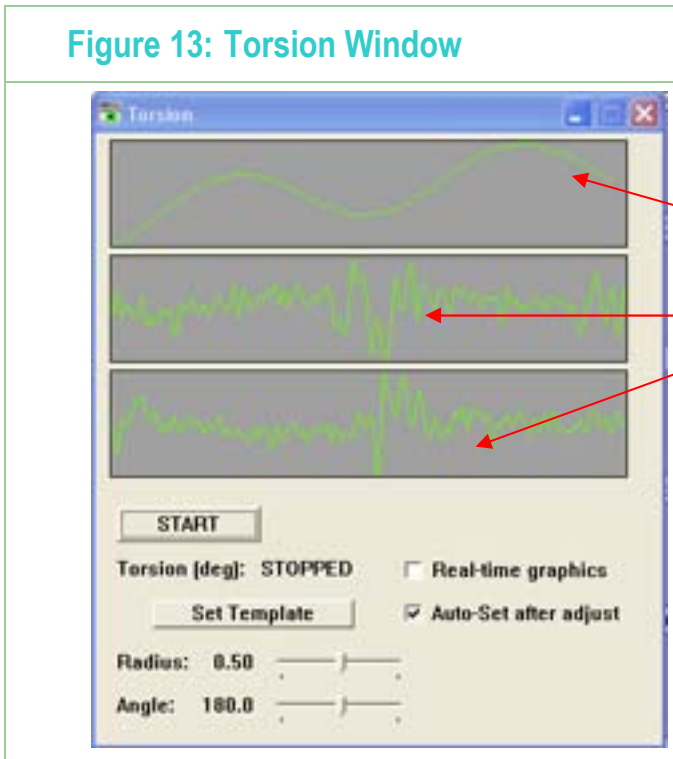
Chapter 8 Ocular Torsion

8.1 Introduction to Torsion

Ocular Torsion is the rotation of the eye ball about the line of sight, i.e. rotation about the z-axis. *ViewPoint* measures ocular torsion by determining the rotation of the iris striation patterns. A representative striation pattern *sample* is stored as the *template* (presumably taken when the eye was at zero degrees torsion). Subsequent samples are compared against the template to determine how much rotation has occurred.

To open the **Torsion** window Select the menu item: **Windows > Torsion**. The **Torsion** window is shown in Figure 13: below. To start torsion measurement, press the **Start** button. When torsion is being calculated, the **EyeCamera** window will contain additional overlay graphics that indicate the circle along which the iris striations are sampled, as well as the starting point on the circle for the sample array. These additional overly graphics are shown in Figure 14: below.

Figure 13: Torsion Window



Graphic well descriptions:

- Correlation between the current sample (middle graph) and the template (bottom graph).
- Current sample of iris striations.
- Samples that were stored as the template.

Figure 14: EyeCamera window with Torsion ON



The samples of iris striations are taken along a circle around the pupil. The radius of the sampling circle is adjusted using the **Radius** slider in the **Torsion** window. The user should adjust the radius to a location where there is good variation in the iris. Regularly periodic variation of the iris striations, as like a sine wave, does not allow identification of rotation beyond the period of oscillation, so an irregular marking is better to track. The location of the sample circle is drawn in the **EyeCamera** window.

Important: fixation is required for accurate measurement of ocular torsion.

The threshold dots are automatically turned off when using torsion. This is important because the threshold dots are painted in the video image before the torsion sample is taken and this can adversely affect the performance of the torsion calculation.

The amount of calculated torsion in degrees is displayed in the **Torsion** window, the **PenPlot** window and stored in the data file

8.2 Procedure for Measuring Torsion

This section describes the procedure for obtaining ocular torsion measurements using *ViewPoint*. It is assumed that the user is familiar with setup and thresholding as described in Chapters 0 and Chapter 5.

Select the menu item: **Windows > Torsion** to display the **Torsion** window.

Ensure **Auto-Set After Adjust** is checked.

Ensure that the subject is in a comfortable position that will allow them to remain still for the duration of the experiment.

1. Press the **START** button on the Torsion window.

Adjust the camera so that the video image of the subject's pupil is in the middle of the **EyeCamera** window and the iris is in sharp focus.

Adjust the brightness and contrast if necessary. Threshold the image.

Instruct the subject to fixate at a given point.

Using the **Radius** slider, adjust the size of the sampling circle to a location where there is strong irregular variation in the iris striations.

Ensure that the sample area circle does not include specular reflections, the eye lid, and any other non-rotating areas of brightness or shadowing.

Press the **Set Template** button, when the subject's eyes are at zero torsion.

Collect data as usual.

Notes: When the Real-time graphics check box is checked, the graphics windows are updated with every new field. To reduce the computational burden, uncheck Real-time graphics. This will cause the graphics to be updated every 30th field. This does not affect the real-time data stored in the data file.

8.3 Torsion Demonstration Test

Normally the sample array starts from the 3 O'clock position on the circle and proceeds to sample pixels along the circle in a clock-wise direction. This starting point can be adjusted using the **Angle** slider. This is shown in the **EyeCamera** window by a line drawn from the center of the pupil to the point on the edge of the circle where sampling begins. (Shown in Figure 17) This can be used for testing and demonstrating the torsion calculation as follows:

- Uncheck Auto-Set After Adjust.
- Adjust the sample angle using the **Angler** slider.

As the pattern shifts away from the template pattern, the correlation peak shifts and the torsion calculation changes.

When not performing this demonstration, **Auto-Set After Adjust** should always be checked, since a new autocorrelation template will be required if the radius of the circle and angular starting point on the circle is adjusted. The autocorrelation template can also be re-fixed manually at any time by pressing the **Set Template** button.

Section 14.17 describes the torsion commands in detail.

8.4 Overriding the Default Torsion Parameters

The default setting is for *ViewPoint* to look for torsion over +/- 9 degrees. Beyond this will cause a "Range Error" to be reported in the **Torsion** window. The default precision is 0.5 degrees of arc. These defaults are in place as a trade off between range of torsion measured and resolution due to the high computational burden of the calculations performed.

Since the eye does not normally rotate about the line of sight more than about 9 degrees there is usually no need to perform the auto-correlation past this range, because increasing the range increases the cpu load unnecessarily. There are some situations in which this range needs to be increased, such as when the entire head is rotated.

The user may adjust the torsion parameters with settings file commands; however, the user is responsible for testing that the combination they choose will provide valid results. Valid combinations should work up to +/-20 degrees at 0.5 resolution.

Chapter 9 Stimulus Presentation (Head Fixed)

This section describes stimulus presentation options using the ViewPoint EyeTracker® PC-60 head fixed (QuickClamp) system or head mounted display (HMD) module systems.

9.1 General

The eye tracker is integrated with the ability to display stimuli. By selecting menu item: **File > Image > Load Image ...** a stimulus picture (BITMAP file) can be chosen using the standard open file dialog box. The picture will appear in both the **GazeSpace** window and in the **Stimulus** window. *ViewPoint* assumes that the BITMAP (.bmp) files are stored in the folder named "**Images**" that is located in same folder as the *ViewPoint* application program. To override this, an alternative full path can be specified via CLP command, see section 14.22.2.

Hint 1: Make the bitmap image large so to avoid smooth lines being displayed as jagged.

Hint 2: Make the bitmap image the same aspect ratio as the **Stimulus** window.

The user has control over how images will be proportioned when they are displayed in the **Stimulus** window and **GazeSpace** window. By selecting menu item **Viewing Source > Image Shape >**:

Actual size: the image will be displayed in the two windows at actual size.

Center: the image will be displayed actual size and centered in the windows.

Stretch to Window: the image will be scaled to fit the window.

Stretch isotropically: the image will be stretched equally in all directions, maintaining the original proportions.

Menu item: **Viewing Source > Background Color** allows the user to change the background color in the **Stimulus** windows. This is useful to provide "matting" color when the user has selected the image shape to be isotropic and there is space at the sides of the sides or at the bottom of the image.

9.2 Picture Lists

A list of picture file names may be loaded by using the settings file commands:

```
pictureList_Init and pictureList_AddName myImageFileName.bmp
```

Refer to Chapter 14.4 PictureList for a full list and description of commands.

After being loaded, this list can be randomized and sequentially presented using the following menu commands:

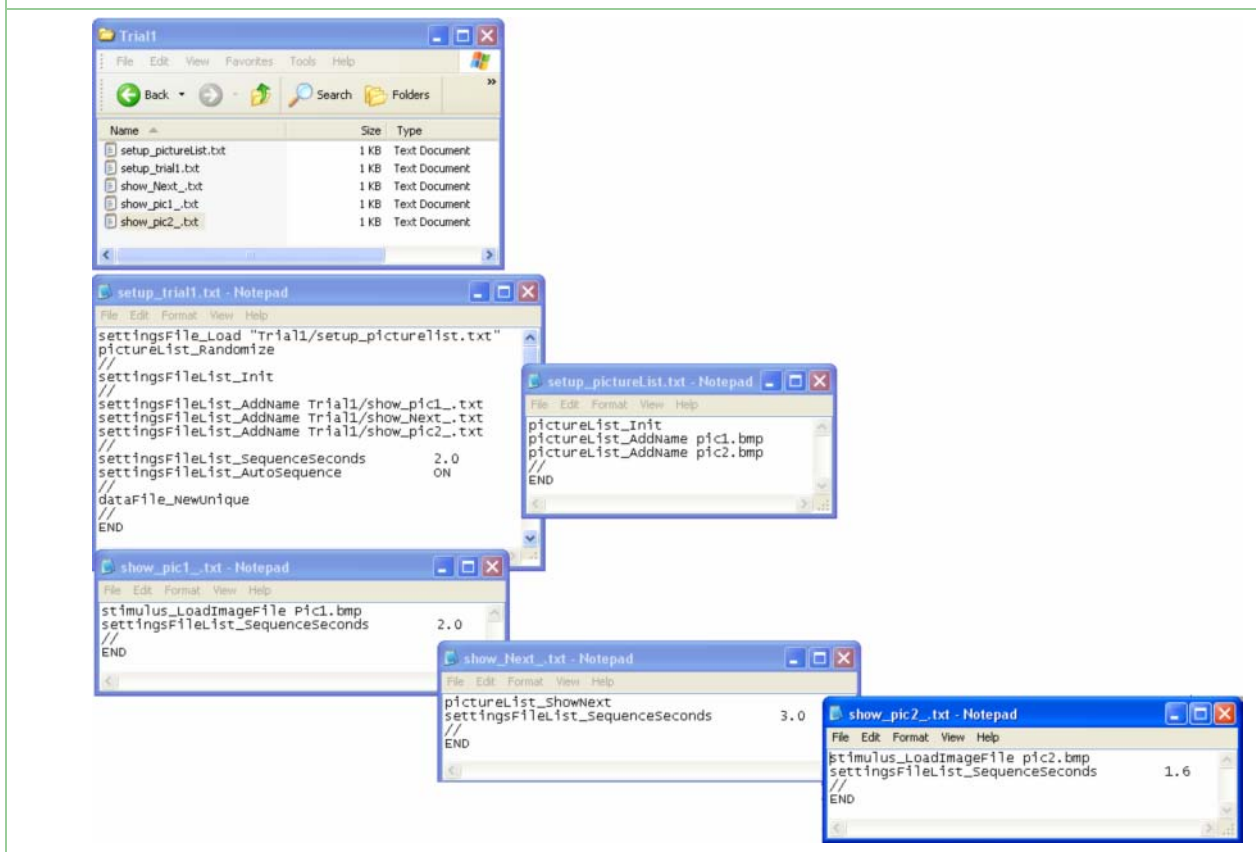
File > Image > Picture List > Next Picture List Image
Presents the next image in the currently setting file list

File > Image > Picture List > Restart PictureList Returns to the top of the currently loaded settings file

File > Image > Picture List > Randomize PictureList Randomizes the list of images in the currently loaded settings file.

The user can create a list of settings files. Each settings file may contain not only commands for displaying picture images, but also for loading unique regions of interest (ROI) for each picture, for playing a cue sound, etc. Figure 15: below demonstrates how settings files can be used very simply to present a series of stimulus images at intervals determined by the user.

Figure 15: Picture List Example



Further examples can be found in the settings folder provided on your *ViewPoint EyeTracker®* software disk.

9.3 Using the Stimulus Window (Head Fixed Option)

The **Stimulus** window is the window the subject sees. Calibration stimulus points and stimulus images are shown to the subject in this window. It is best when displayed full screen on a second monitor!

Use the **Controls** window **DataDisplay** tab to remove or show the stimulus picture image. When unchecked the image is removed and all you see is the plain background color selected by the user.

- **Stimuli > View Source > Stimulus Window Properties > Normal Adjustable**
Makes the **Stimulus** Window a resizable, moveable window.
- **Stimuli > View Source > Stimulus Window Properties > Custom Static Position**
Sets the **Stimulus** Window to be a custom size as specified by the currently loaded settings file. This window is not resizable or moveable. This feature is for use with non-standard display cards.
- **Stimuli > View Source > Stimulus Window Properties > Monitor 1 (primary)**
Sets the **Stimulus** Window to be full screen on the primary monitor.
- **Stimuli > View Source > Stimulus Window Properties > Monitor 2**
Sets the **Stimulus** Window to be full screen on the secondary monitor

Note: To use a second monitor you will need to install a second display card into your computer and configure your computer for multiple monitors. Please refer to your computer manual.

- **Stimuli > Stimulus Window Properties > AutoShow on calibrate** (toggle)
automatically shows the **Stimulus** window and also automatically hides the cursor (only during calibration, re-present or slip correction) when the AutoShow occurs.

9.4 Using the GazeSpace Window

The **GazeSpace** window is a miniature representation of the **Stimulus** window. The experimenter may select from a number of ways to show and view the subject's instantaneous position-of-gaze, using the check boxes on the **Controls** window as follows:

- **Calib Region** shows the area within which the calibration is performed, see section 6.12.3, (unless custom calibration stimulus points are enabled, see sections 6.13) and the locations of the calibration stimulus points.
- **Gaze Point** shows the subjects position of gaze.
- **Trace Lines** shows the path of eye movements.

- **Fixation Time** displays fixation duration as the area of a circle increasing as the duration increases.
- **Pupil Size** will display a yellow oval corresponding to pupil size at the position of gaze.
- **ROI Regions** displays the region of interest boxes.
- **GeometryGrid** to display the GeometryGrid. For GeometryGrid setup details refer to (6.4)
- **Raw Data** to display the raw, uncalibrated data.
- **Picture Image** to display the currently loaded stimulus picture image.

Eye movement traces are usually only presented to the experimenter in the **GazeSpace** window, but they can also be presented to the subject in the **Stimulus** window. Presentation of gaze information is controlled through the **Controls** window. Display of eye traces in the **Stimulus** window is useful during setup and self testing, but is not recommended during normal operation, since they can be very distracting to the subject. The experimenter may select from a number of ways to show and view the subject's instantaneous position-of-gaze.

9.5 Regions of Interest (ROI)

The stimulus area can be divided up into regions of interest (ROI, also sometimes called areas of interest (AOI), or window discriminator boxes. These are very useful when the experimenter wants to know categorically whether or not the subjects gaze was in a certain area.

It is possible to specify up to 100 ROI boxes (box numbers 0-99) to simplify the task of data analysis. When the gaze position moves inside a ROI, the ROI box number is displayed in the **Status** window and stored in the data file record if a data file is open. If the boxes are overlapping and the gaze is inside multiple boxes, then both the **Status** window and the data file will list all of the ROI boxes that were "hit".

To adjust individual ROI, check the Region of Interest radio button on the **Controls window: Regions tab**. Individual ROI may be selected by clicking the right mouse button inside the ROI. The selected ROI will be drawn in red and the others in blue. Use the left mouse button to move and resize the selected ROI. Clicking the right mouse outside of any ROI will set the adjustment mode to the locked state. Region boxes can also be made active and locked through the **Regions** tab on the **Controls** window. Use the slider to sequence through the RIO. Use the mouse wheel to sequence through and add new ROI. The active ROI coordinates are displayed and updated as you adjust the size and position both in the GazeSpace window bar and the **Regions** tab. The **Revert** button in the **Controls** window **Regions** tab will undo the last change and the Default button will return the ROI boxes to the start-up setting.

9.6 Data Smoothing

The eye trace lines displayed in the **GazeSpace** and **Stimulus** windows can be smoothed to reduce noise. The degree of smoothing of gaze calculation may be varied using the slider on the **Controls** window (see Figure 9). When placed at the far left, no smoothing is performed. Incrementing the slider to the right increases the number of previous points included in the average calculation. A value of 4 makes attractive and useful real-time graphics. Two smoothing algorithms can be used, in each case the number of back points equals the number set by the user. Refer to Section 11.1.1.

Smoothing effects the real-time calculations. Because smoothing effects the velocity calculations the saccade velocity threshold must be adjusted proportionately. Smoothing will also affect which ROI boxes are triggered.

By default, smoothing does **not** influence the data values that are stored to file, because the real-time smoothing uses a trailing average technique, whereas post hoc data analysis should use a symmetrical smoothing technique. The user can choose to store the smoothed data by selecting menu item **File > Data > Store Smoothed Data**.

Note: For post-hoc analysis it is preferable to use a symmetric smoothing kernel on unsmoothed data.

9.7 Using the SDK, settings files and Serial Port Interface for Stimulus Presentation

There are two ways to control the *ViewPoint EyeTracker*[®] from other applications and other computers:

Programming functions – handled by the SDK in the DLL

Command strings – handled by the Command Line Parser (CLP) in the *ViewPoint* application.

The same CLP is used for command strings received from:

- *Settings* files that are loaded.
- serial port packets of type COMMAND_LINE.
- The SDK using the VPX_SendCommand function.

The SDK is described in Chapter 15, Settings Files in Chapter 12, and the Serial Port Interface in Chapter 13. Chapter 14 lists all of the available ViewPoint features, the GUI control, CLP command and any applicable SD function. There are many features available through CLP commands or SDK functions only.



9.8 Integrating with Third Party Products

The *ViewPoint EyeTracker*[®] has been integrated with many of third party experiment generation software products. Please contact us for the latest information.

Chapter 10 Data Collection

10.1 Sampling Rate

The *ViewPoint EyeTracker*[®] includes three modes of operation that provide flexibility in the selection of resolution and sampling rate. Which mode you choose will depend on your research or project requirements. However, you should always calibrate using the [High Precision Mode](#).

- **Setup Mode:**
Temporal Resolution: 30Hz
Internal Processing: 320 x 240
- **High Precision Mode:**
Temporal Resolution: 30Hz
Internal Processing: 640 x 480
- **High Speed Mode:**
Temporal Resolution: 60Hz
Internal Processing: 640 x 240

Menu Item: [Video > Mode > *](#) can be used to select the required operating mode. The icon button on the [EyeCamera](#) window tool bar can also be used to sequence through the operating modes ([SetUp](#) – [High Precision](#) – [High Speed](#) – [SetUp](#)). Historically, when computers were slower, the image of the eye only appeared in [Setup Mode](#); the other modes only displayed the overlay graphics, to save cpu time. This is no longer the case and the term is somewhat misleading because “setup” can be done in any of the modes. This mode may better be described as a “low cpu” mode, nevertheless, the old terminology is retained for the present.

10.2 Saving Data to File

The ViewPoint data file is always saved as tab-delimited text, sometimes referred to as the standard spreadsheet format, so it can be easily read by most any program, e.g. MS Excel, MS Works, Matlab, Mathematica, etc. The file extension does not affect the internal format of the data file. You can specify, or change the file extension whenever, and to whatever you want (e.g.: “.txt”). You can associate the file extension with specific applications at the operating system level, so that when you double-click the file icon, a specific application will open it.

The data file will contain information about the (x, y) gaze point, elapsed time since the last entry, total time, pupil diameter, and a region of interest box number, etc. By default, the data files are stored in the folder named “**Data**” inside the *ViewPoint* application folder. The default folder can be change using command line instruction [setPath](#), described in section 14.22.2.

To start recording data to file either:

- Select menu item: [File > Data > New data file....](#) This allows the user to open a

data capture file, and to specify a file name using the standard *Open File* dialog box. Or,

■ Select menu item: **File > Data > Unique data file ...** This opens a new data capture file with a unique file name without having to go through the standard *Open File* dialog box. The default extension for these files can be set using the CLP command: `dataFile_NewUniqueExtension` described in section 14.2.2.

File > Data > Pause Data Capture will temporarily stop the recording of data to file while processing continues. Data recording can be continued by toggling this menu item off. This feature is different from menu item **Video > Freeze** which stops all processing and recording of data to file. **File > Data > Close Data File** terminates the writing of data and closes the open file.

The command line interface for data files is described in section: 14.2.

10.3 Data File Format

10.3.1 File header information

At the top of each data file is file header information that includes the data and time that the data was collected, the apparatus geometry settings that can be used to obtain the gaze angle in degrees, whether smoothed or unsmoothed data was stored, etc.

10.3.2 File records

Each line of the data file is a unique data record. The type of record is indicated by the integer “*Tag*” value in the first column. The tag lets us know how to interpret the entries that follow on that line. The record entries on the line are tab separated into column positions. The number of columns in the eye-data record will depend upon the options selected for data collection. For example, if torsion is not being measured there will be no such column in the data file. If running in binocular mode, then additional data will be included for the second eye. The meaning of the various data record tags is described in detail in section 10.3.4.

10.3.3 Synchronous vs. Asynchronous data inserts

ViewPoint provides for synchronizing eye movement data with other events and other data. This is usually performed by inserting extra data into the *ViewPoint* data file. This extra data can include Markers, Strings, and HeadTracker data. These can be inserted in two ways, either (a) on the same line as the eye tracker data (synchronously), which makes reading into spreadsheets much easier, or (b) on separate data lines (asynchronously), interleaved with the eye tracker data, which allows individual time stamps for each inserted item.

A simple method of synchronization with other devices and programs is achieved through insertion of ASCII character markers into the data stream. By default, menu item **File > Data > Asynchronous Marker Data** is toggled OFF, so data markers will be synchronously added into the data file stream, in the Marker column. If more than one character was inserted, they will appear as a comma separated list of characters. If this menu item is checked ON, ASCII character data markers will be asynchronously inserted into the data file stream as they arrive, with individual time stamps, and there will be no Marker column.

By default, menu item **File > Data > Asynchronous String Data** is toggled ON and string markers

will be asynchronously added into the data file stream with individual time stamps.

For example, the following set of instructions:

```
dataFile_asyncStringData YES
dataFile_InsertString "CAT_A (asynchronous)"
dataFile_InsertString "CAT_B (asynchronous)"
dataFile_asyncStringData NO
dataFile_InsertString "DOG (synchronous)"
```

produce the following data file lines:

```
10 8.8630 33.3186 0.6464 0.4391 -1 0.1048 0.5589 0 0.5330 265
10 8.8963 33.3219 0.6464 0.4391 -1 0.1048 0.5589 4 0.5330 266
12 8.946231 CAT_A (asynchronous)
12 8.946242 CAT_B (asynchronous)
10 8.9296 33.3189 0.5263 0.4864 -1 0.0829 0.4728 0 0.0831 267 DOG (synchronous)
10 8.9630 33.3235 0.5263 0.4864 -1 0.0829 0.4728 4 0.0831 268
```

10.3.4 Data Record Tags.

The “tag” value that begins each line in the first column indicates the type of information in that line, i.e., in that data record. There are seven record types, as follows:

Tag #10: EyeData containing a variable number of columns depending on the options selected for data collection. Table 3

Tag # 2: ASCII Character Marker. If menu item **File > Data > Asynchronous Marker Data** is checked ON, then single ASCII character event markers will be inserted asynchronously into the data file. Certain ASCII characters are automatically entered into the data stream to indicate a particular event has occurred. The type-2 record contains three entries, as described in Table 4

Tag # 12: An ASCII character string asynchronously inserted during certain events. For example, by the CLP in response to certain commands.

E.g. “pictureList_ShowNext.”, “dataFile_InsertString picture of a cat” from a settings file, or an external program link (serialPort or Ethernet packet). [Table 7:](#)

Tag # 3: An ASCII character string generated by *ViewPoint* to provide general information, such as when the data file was created. Refer to Table 6

Tag # 5: An ASCII character string generated by *ViewPoint* to provide column heading information. Refer to Table 8.

Tag # 6: A 3 character data column identifier generated by *ViewPoint*. Refer to [Table 8.](#)

Tag # 14: Asynchronously inserted head tracker data.

Note: The user can insert uniquely tagged data from their own sources into a *ViewPoint* data file using the CLP command `DataFile_InsertUserTag`. The insertion is done asynchronously with respect to the eye movement data records and the insertions are uniquely time stamped.

Table 3: DataFile: EyeData Record Structure, Tag = 10		
<i>Column Heading</i>	<i>Type</i>	<i>Description</i>
Tag	integer	The value 10 in the first column indicates an eye data record.
TotalTime	float	TotalTime = time elapsed in seconds
DeltaTime	float	dt = delta time in milliseconds since the previous data entry
X_Gaze	float	X = direction of gaze normalized with respect to the x-axis
Y_Gaze	float	Y = direction of gaze normalized with respect to the y-axis
Region	list	Which ROI or ROIs the gaze point is in
PupilWidth	float	Pupil width normalized with respect to the EyeCamera window width
PupilAspect	float	Dimensionless aspect ration of the pupil, i.e. 1.0 is a perfect circle
Quality	integer	Quality of eye movement data. See section 10.7, for a complete description of the codes.
Fixation	float	Fixation duration in seconds. A zero value indicates a saccade.
Torsion	float	Torsion in degrees. -998 indicates Torsion not being calculated. -999 indicates "Range Error". Only displayed if torsion is being measured.
Count	integer	Eye movement data record count, useful for sorting.
Mark	char	Any ASCII character, e.g., {a-z, A-Z,0-9,=,#,+,%, etc.}. See Table 4:, below.

Table 4: Asynchronous Marker Record Structure, Tag = 2		
Column #	Type	Value
1	integer	2 = integer indicates data structure type 2
2	float	Time Stamp
3	character	Any ASCII character, e.g., {a-z, A-Z,0-9,=,#,+,%, etc.} Automatically inserted tags include: + Start or resume data collection, and = Pause data collection

Table 5: Asynchronous HeadTracker Data Record Structure, Tag = 14		
Column #	Type	Value
1	integer	14 = integer indicates data structure type 14
2	float	Time Stamp
3	float	Head position and angle data. With head tracker option only. HPX, HPY, HPZ, HAX, HAY, HAZ

Table 6: String Data Record Structure, Tag = 3		
Column #	Type	Value
1	integer	3 = integer indicates data structure type 3
2	string	File header information, an ASCII character string generated by <i>ViewPoint</i>

Table 7: String Data Record Structure, Tag = 12		
Column #	Type	Value
1	integer	12 = integer indicates data structure type 12
2	float	Time stamp
3	string	ASCII character string from "dataFile_InsertString", etc.

Table 8: Column Header Data Record Structure (Tag # 5 and 6)							
Column #	Type	Value					
1	Integer	5	6	Eye			
2 – n	String	Total Time Delta Time X-Gaze Y-Gaze ROI Pupil Width Pupil Aspect Quality Fixation	ATT ADT ALX ALY ARI APW APA AQU AFX	Eye_A			
		Total Time Delta Time X-Gaze Y-Gaze ROI Pupil Width Pupil Aspect Quality Fixation	BTT BDT BLX BLY BRI BPW BPA BQU BFX		Eye_B		
		HPX HPY HPZ HAY HAX HAZ	X Y Z Yaw pitch roll			Head Tracker Option only	
		Record Count	CNT				
		Marker	MRK				

10.4 Direction-of-gaze Coordinates

The values **X** and **Y** are the coordinates of the direction-of-gaze with respect to the **Stimulus** window coordinate systems. For example, 0.0, 0.0 will mean that the position of gaze is in the top left hand corner of the **Stimulus** window; 0.5, 0.5 will mean that the position of gaze is in the center of the **Stimulus** window; and 1.0, 1.0 means that the position of gaze is in the bottom right hand corner.

It is possible to display the eye-traces in the **GazeSpace** window as averaged position that

is with data smoothing. Refer to 8.6 Data Smoothing. By default, the smoothing of the data in the *GazeSpace* window display does **not** influence the XY data values that are stored in the data file (but the ROI box numbers are triggered by the smoothed eye traces). However, the user can choose to store the trailing average smoothed data in the data file using menu item [File > Data > Store Smoothed Data](#).

The calculated (x, y) gaze location is initially in the same space as the calibration point locations. The calculated gaze is then normalized with respect to the x-axis and y-axis respectively. *ViewPoint* allows the GazePoint to be extrapolated outside of the calibration window. This is particularly important for using the CursorControl Feature. (see 6.9 Cursor Control Feature)

10.5 Timing Measurement

The *ViewPoint EyeTracker*[®] includes high precision timing (HPT) with resolution in the order of 0.0000025, i.e., 2.5E-6 or 2.5 microseconds. The HPT is available to integrators via the SDK function call: `VPX_GetPrecisionDeltaTime` Refer to 13.6 for details.

Total Time is the elapsed time in seconds starting with the first record of data collection. By default Menu item [File > Data > Start Data File at Zero](#) is checked (ON) which causes the first record of the data file to start at time zero. If this menu item is unchecked (OFF), the data records will start at the current High Precision Time maintained in the DLL and sharable between applications using the *ViewPoint* SDK via that DLL.

The value **Delta Time** is the number of milliseconds since the previous data stream entry. This represents the time that the software has finished processing each frame and the eye movement data is available.

10.6 Region of Interest (ROI)

Often one is only interested in whether the direction-of-gaze is at a specific location, or more generally, which of several possible locations towards which the gaze is directed. Regions of Interest may be defined as described in 8.5. When the calculated (possibly smoothed) gaze position is within one of these boxes, the region number is stored in the data file. The ROIs may overlap and so the gaze can be in more than one region at a time.

- -1 indicates that the position of gaze (POG) is not in any of the regions,
- n indicates that the POG is in region n,
- n,m indicates that the POG is in more than one, overlapping ROI.

10.7 Quality Marker Codes

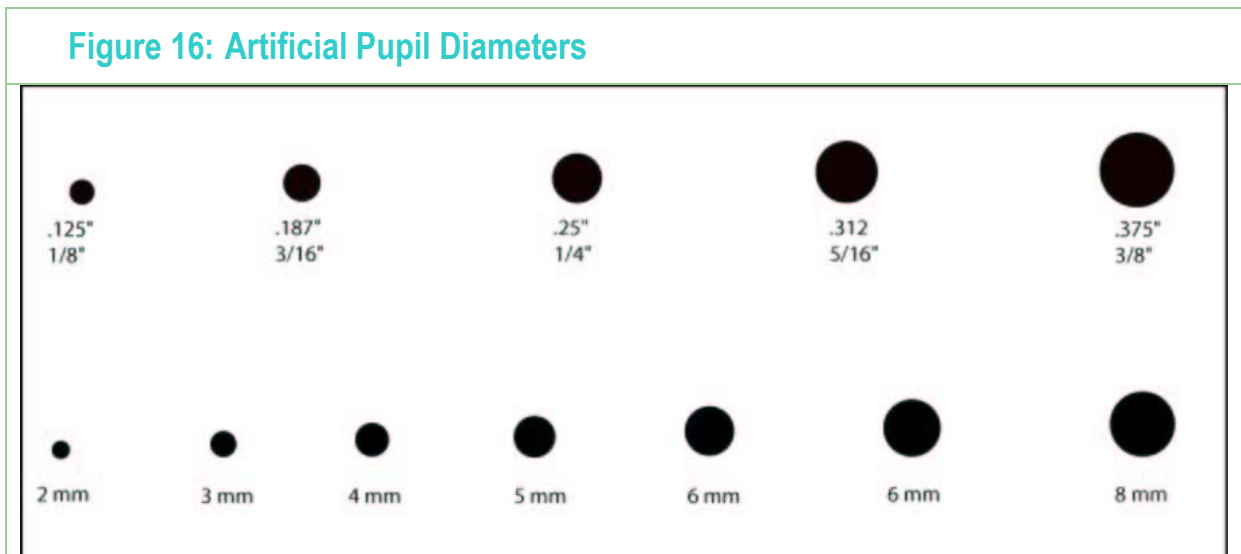
ViewPoint provides a data quality code for each source of recorded data. The Quality column contains an integer code ranging from 0, which is the best possible case, to 5. The hierarchical code structure allows the user to make their own assessment of whether the data is

valid or not for their purposes. Quality marker codes are listed below:

Table 9: Quality Codes	
Code	Description
0	The user has selected to use the <i>glint-pupil vector</i> method and both features are successfully located.
1	The user has selected to use the <i>pupil only</i> method and the pupil was successfully located
2	The user has selected to use <i>the glint-pupil vector</i> method but the glint was not successfully located. Defaults to <i>pupil only</i> method for data recorded.
3	In either the <i>pupil only</i> or <i>glint-pupil vector</i> method, the pupil exceeded criteria limits set.
4	In either the <i>pupil only</i> or <i>glint-pupil vector</i> method, the pupil could not be fit with an ellipse.
5	In either the <i>pupil only</i> or <i>glint-pupil vector</i> method, the pupil scan threshold failed.

10.8 Pupil Diameter

An oval is fit to the located pupil. Pupil oval width and height is in pixels. Obviously the actual pupil diameter will depend on the camera placement relative to the eye. Table 10 Artificial Pupil Diameters contains a set of black disks of specific diameters that can be used as “artificial pupils” for determining what actual pupil diameter (in inches or millimeters) the diameter in pixels corresponds to. Pupil diameters usually range between 2mm and 8mm.



10.9 Pupil Aspect

Blinks can be detected by monitoring the pupil aspect ratio. This is a dimensionless value, where 1.0 indicates a perfect circle.

10.10 Display Screen Geometry

The data file also contains header information that specifies the results of the [GeometryGrid](#) calculations for screen size and viewing distance. The screen size values saved will be the values calculated for the entire screen width and height, not the measured lengths of the lines (that are 20% less) entered by the operator.

Chapter 11 Data Analysis

11.1 Real-Time

The *ViewPoint EyeTracker*® provides many means of monitoring eye movements in real-time including the *PenPlot*, *GazeSpace* and *Status* windows.

Figure 17: Status and GazeSpace Windows

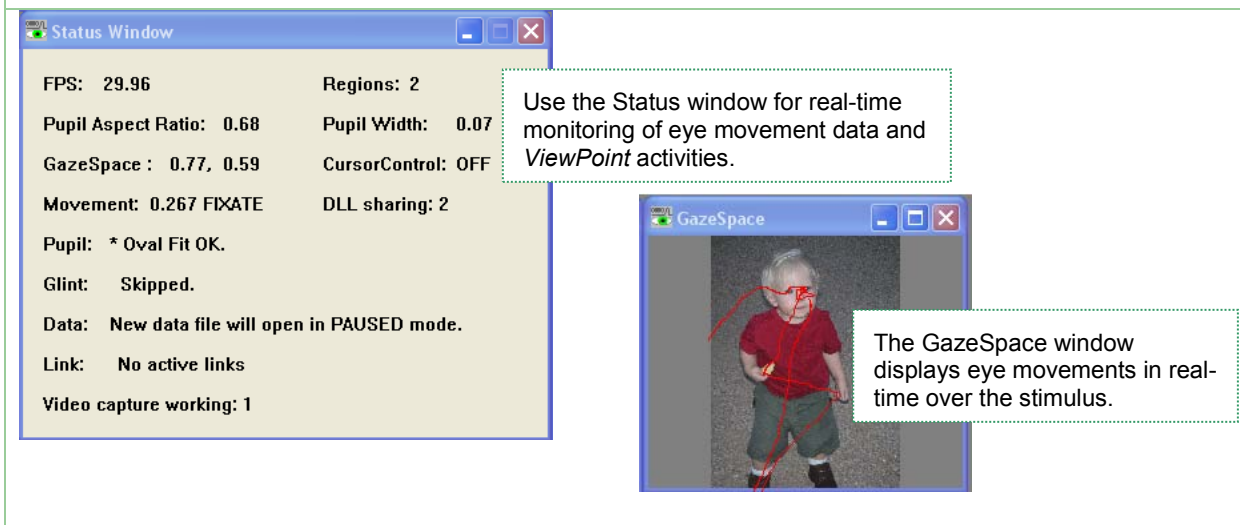
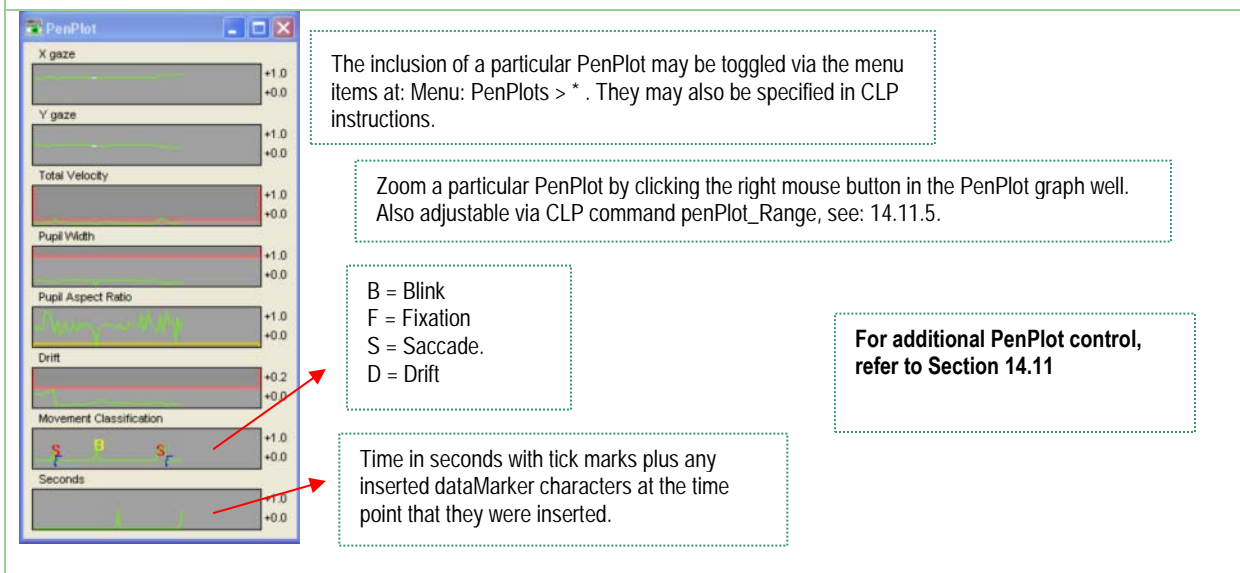


Figure 18: PenPlot Window



11.1.1 Data Smoothing

Data displayed in the **PenPlot** and **GazeSpace** windows may be smoothed. The user can select the amount of smoothing using the slider on the **Controls** window **DataCriteria** tab or with CLP commands. Two smoothing algorithms can be used, in each case the number of *pointsBack* equals the number set by the user:

Simple Moving Average (SMA).

The SMA method uniformly averages N *pointsBack*, i.e., all points having equal weight.

$SMA(t) = [x(t) + x(t-1) + \dots + x(t-n)] / N$; where $n = (N-1)$

Exponential Moving Average (EMA).

The EdMA method uses the following algorithm:

$EMA(t) = (\text{currentValue} - EMA(t-1)) * K + EMA(t-1)$; where $K = 2 / (\text{pointsBack} + 1)$.

NOTES:

SMA is the default setting.

The data is stored UNSMOOTHED unless specified by the user.

The successfully set method is currently reported in the Event History window and the Status Window.

11.2 Fixation, Saccade, Drift and Blinks

11.2.1 Velocity Threshold

The **Saccade Velocity** slider on the **Controls** window **DataCriteria** tab provides a qualitative means for discriminating saccades from fixations with noise. Adjustments to this threshold should be done while examining the **Total Velocity** in the **PenPlot window**. You can see when the subject is fixating and when they saccade from the spikes in this trace. The value of the spikes is simply the change in 2D position without begin divided by sampling interval time.

Looking at the **Penplot** when the subject is fixating you can see that the **Velocity** trace contains noise. (thermal, video noise etc.) If saccades are large, then the placement of the saccade threshold is not so critical. If saccades are small more care should be taken. There is a trade-off in terms of misclassifying noise as a saccade or a small saccade being below threshold are missed.

11.2.2 Fixations

ViewPoint calculates fixation as the length of time that the velocity was below the saccade velocity criterion. Fixation duration is included in the data file as a cumulative value, is displayed as a value in seconds in the **Status** window, as a ramp function in the Fixation Time pen plot, and the fixation start time is indicated in **Events** plot by the letter 'F'. At the request of several customers, we have included the fixation duration value into the data file (in the AFX column). This

are the same value as appear in the [Status](#) window and is plotted in the [PenPlot](#) window in real-time, based on a trailing average smoothing algorithm and the qualitative saccade velocity threshold adjusted by the user (possibly during data collection). It may be preferable to apply a symmetrical smoothing kernel to the un-smoothed data during post-hoc data analysis and to determine the optimal saccade criteria at that time.

The velocity value is simply the difference between the current and the last calculated and smoothed gaze points, i.e., the change in the normalized position of gaze, – hence smoothing will affect the velocity magnitude.

11.2.3 Drift

Note that the eye may be slowly drifting or in a low velocity smooth pursuit, such that the velocity is below the [Saccade Velocity](#) criterion. For this reason we also provide a [Drift](#) criterion that establishes the maximum distance that the gaze point may move away from the putative fixation point. That is, we impose the additional criterion that the position of gaze must remain within a certain distance of the initial fixation point (or the previous drift point). Fixation drift is in normalized screen coordinates as the calculated position of gaze.

```
GUI: Controls window : DataCriteria tab : Drift Allowed slider  
CLP: driftCriterion normalValue
```

The absolute drift distance and the specified Drift criterion level can be visualized in the Drift penPlot well as the letter 'D'.

```
GUI: menu Windows > PenPlots > Drift  
CLP: penPlot +DRIFT
```

11.2.4 Blinks

As the eye lid comes down during a blink, the elliptical fit to the pupil becomes increasingly flat before it disappears. This characteristic change in the aspect ratio of elliptical fit to the pupil can be used to detect blinks. A blink is classified as the pupil aspect ratio crossing below the threshold.

11.2.5 Events

Fixation, Saccade, Drift and Blink events are displayed in the Events penplot graphics well as the following characters:

- F – fixation
- B – blink
- S – saccade
- D – drift

11.2.6 SDK

The real time values for all data points are available via the DLL based SDK. See section 15.8.

11.3 Post-Hoc

The data file format is described in Chapter 10.

We also provide a basic data analysis program that allows displaying data in both a time plot and a 2D (x,y) plane overlaid on the visual stimulus. The *DataAnalysis* program can be launched by itself from the folder or it can be launched when the data file is closed.

Menu: File > Data > Close & Analyze Data File

CLP: dataFile_CloseAndAnalyze

Chapter 12 Using Settings Files

All the graphical user interface (GUI) controls in the ViewPoint application have equivalent command line interface (CLI) strings that are interpreted by the command line parser (CLP). All of the GUI values and selections (menu selections, slider values, etc.) can be saved in a *Settings* file that can be later read back into the CLP next time the program is run, so the user can start working without resetting everything by hand. The *Settings* file also contains calibration values, region of interest (ROI) specifications and other program variables. By default, *ViewPoint* assumes that the *Settings* files are stored in the folder named "*Settings*" that is located in same folder as the *ViewPoint* application program.

The *Settings* files are in (usually tab delimited) ASCII format. Settings files can be saved and loaded using menu selections, as described in 10.1 *Saving and Loading Settings Files*.

The *Settings* file consists of individual lines of ASCII text that may be edited using an editor. The document must however (a) be saved as text only (b) have each command separated by a semi colon, and (c) conform accurately to spelling and spacing requirements. The CLP is not case-sensitive. An editor with good tab setting capabilities is recommended because the row entries are tab separated. The default settings file extension is .txt which helps exclude unusable files that contain extra formatting, such as rich text format (.rtf) and *MS Word* format files. Use menu item **File > Settings > Edit Settings** to select an existing settings file to edit.

Do not modify anything that you do not understand.

Important Notice: these commands (names and argument lists) may not be consistent with the Macintosh ViewPoint settings, and they are subject to changes in future versions. Note: The strings are not case sensitive.

Settings Files may be nested (i.e. one *Settings* file may call another *Settings* file, see: `settingsFile_Load`). Consequently, it is desirable for each file to be reasonable in length. Currently the *Settings* file is limited to 800 lines, including comment lines. Note however that the same settings file cannot be called recursively. *Example settings files are contained in the Settings folder on the ViewPoint EyeTracker® disk supplied with your system.*

12.1 CLP String Parsing

All white spaces are gobbled up until the beginning of a string is specified by either (i) a non-white space character, or (ii) a beginning quote is encountered. Quotes should be matched (this was not expected in previous versions). An inline double-forward-slash will cause everything remaining on the line to be ignored as a comment.

12.2 Saving and Loading Settings Files

File > Settings > Load Settings
allows the user to read in the settings, using the standard open file dialog box.

File > Settings > Save Settings...

allows the user to store the current Settings to file, using the standard open file dialog box.

File > Settings > Verbose Loading

causes additional information from the CLP to be displayed in the **Event History** window.

File > Settings > Save Window Layout

saves the size, location and z-ordering of all ViewPoint windows.

12.3 Pre-load Settings in a Startup file

When *ViewPoint EyeTracker*[®] is launched it loads in the content of the file `startup.txt` that is located in the folder named "Settings". This can be used to load regularly used settings to reduce setup time.

12.4 Settings/LastRun.txt

ViewPoint automatically creates the settings file `Settings/LastRun.txt` upon quitting. The user may reload these manually at any time, or the user may edit the "Settings/Startup.txt" file to include the following command that will automatically load the previous settings whenever *ViewPoint* is restarted: `settingsFile_Load LastRun.txt`

12.5 Settings File Lists

A simple sequential state-logic is provide by allowing the user to specify a list of settings files and allowing a variable time delay before loading the next file in the sequence. The **Settings File List** to be sequenced through may be set up using a group of CLP command (`settingsFileList_Init` , etc.) described in section SettingsFileList commands 14.16. Start and control the sequencing via the menu item **File > Settings File > SettingsFileList**. *Hint:* It is useful to assign FKey commands for these.

12.6 SettingsFile Examples

It is often a good idea to create individual settings files for different groups of related commands, and then call that settingsFile from a main settingsFile.

Example 1: create individual settingsFiles that contain the name of a bitmap image and the ROIs for that image.

File: imageAndRoi_1.txt

```
stimulus_LoadImageFile picture1.bmp
setROI_RealRect 1 0.1 0.1 0.3 0.2
setROI_RealRect 2 0.4 0.4 0.5 0.5
```

File: imageAndRoi_2.txt

```
stimulus_LoadImageFile picture2.bmp
setROI_RealRect 1 0.6 0.1 0.9 0.2
setROI_RealRect 2 0.4 0.7 0.5 0.9
```

File: startup.txt

```
settingsFile_Load imageAndRoi_1.txt
```

Example 2: create individual settingsFiles that set the FKEY commands for a particular task

File: fkeysForCalibration.txt

```
fkey_cmd 9 calibration_selectPrevious
fkey_cmd 10 calibration_snap
fkey_cmd 11 calibration_selectNext
```

12.7 CLP Commands

The same Command Line Parser (CLP) is used for command strings received from:

- **Settings files** that are loaded.
- The SDK using the **VPX_SendCommand** function (including RemoteLink)
- Serial port packets of type COMMAND_LINE.

The total command line length should not exceed 255 characters. See Chapter Chapter 12 for details about using CLP commands.

12.8 Associating CLP Commands with FKeys

CLP commands can be associated with FKeys. These associations can be viewed in the Info panel: menu Help > Info > **ShortCuts** tab. Refer to 14.23.1

Chapter 13 Serial Port Communication

The *ViewPoint EyeTracker*[®] provides real-time communication with other computers via a serial interface. This data includes (x, y) position, region of interest (ROI) box numbers, pupil size, and timing data, calibration point locations and events, window discriminator box location changes, etc., as described in Chapter 9

13.1 Getting Started

Make sure that you are using a cable that is designed to connect two computers together (cross-over cable), rather than a computer and an external device. This is a cable that switches the send and receive wires so that the send from one computer goes to the receive of the other computer. (not straight through)

You may want to write your own interface program that uses the serial information, but to get started, you can test serial data transfer by using a copy of the *RemoteLink*[™] program on the receiving computer. Select menu item **Interface >** and ensure that the correct port (the one your serial cable is plugged into) is selected and the baud rate matches that selected on the interfacing application. Data bits "8", parity "None" and Stop Bits "1" are a standard configuration and should be changed only if required to match those of the interfacing application. We recommend trying flow control with DTR/DSR enabled. XON/XOFF should always be disabled.

ViewPoint does not by default, monopolize the serial port, so it is necessary to connect to the serial port by selecting menu item **Interface > Serial Port > Connection Settings**. After you have finished with it you should disconnect so that other applications can use the serial port. If the selected port is in use by another device, an error box will be displayed.

13.2 Sending Real and Test Data

To transfer real-time eye data, select **Interface > Serial Port > Send Streaming Data**.

The menu item **Interface > Serial Port > Send Test Pattern** will send artificial position-of-gaze information, and other information, that forms a geometric pattern, which can help in testing and demonstrating the serial communication feature.

Menu Item **Interface > Serial Port > Send Events Only** if toggled ON will send serial port data only when there is a significant event. Significant events are specified as:

a saccade or fixation

an entry or exit from any of the first 10 ROI.

1. drift

To view the events as they occur in real-time, enable **Seconds** in the penPlot window.

Use the CLP command `EventFilterOptions` to control what constitutes an event.

`eventFilterOptions +saccade +roi -drift`

Menu Item **Interface > Serial Port > Send Single Packet** if toggled ON will send a single packet of serial data.

Note: These can be controlled from the remote computer if eye data is only needed occasionally.

13.3 Transfer to Intel and Macintosh Machines

ViewPoint uses **Big Endian** data storage (also called **Network-Endian** because it is the standard for the internet). A Big Endian machine stores the hexadecimal value 0x1234 as (0x12 0x34), while a Little Endian machine stores the same value as (0x34 0x12).

You are all set if you are receiving the data packets on an Apple™ Macintosh™ computer, because Big Endian is the native data storage format for this host. This is true for both the traditional Motorola™ and the newer IBM™ processor machines.

If you are receiving the data packets on a Windows™ PC using an Intel™ processor, then you will need to swap the two bytes composing the unsigned short integers (UINT16), because Intel machines use Little Endian data storage.

13.4 Connections

Newer computers do not have external serial ports, but instead have USB ports and FireWire ports. Serial data can nevertheless still be sent by using a 3rd party USB-to-Serial adapter or by using an add-on PCI card with a serial port.

13.5 Serial Protocol

It is necessary to match the serial port settings of the received program to those of the sender program.

Table 10: ViewPoint Default Serial Port Settings	
Setting	Value
Baud rate	56K (57,600)
Data bits	8
Parity bits	0 (no parity)
Stop bits	1 (one stop bit)

13.6 Serial Packet Header Structure

Note carefully that *ViewPoint* uses a **packet** scheme to transfer data. This means that the bytes are not ascii data and **should not be displayed using a terminal emulator program**.

The serial information is sent in bytes without any intrinsic word alignment. A delayed, dropped or corrupted byte can cause misalignment of multi-byte information, such as integers. Consequently, it is desirable to verify that a byte sequence starts with a valid Packet Header before reading in what is assumed to be data. The Packet Header consists of four bytes as follows:

Table 11: Serial Packet Header Contents		
Setting	Type	Value
1	byte	Synchronization Byte-1, corresponds to ascii char 'V', decimal 86 (hex 0x56)
2	byte	Synchronization Byte-2, corresponds to ascii char 'P', decimal 80 (hex 0x50)
3	byte	SerialTag Byte, numerical value indicates the type of packet that follows.
4	byte	PacketSize Byte, numerical value is the number of bytes in the packet body.

The receiver program should verify that the first byte is the character 'V' and that the second byte is the character 'P', indicating that the data is from the *ViewPoint* program.

13.7 Serial Packet Data Structures

There are several types of packets that contain different types of information.

Table 12: Packet Types			
<i>Direction</i>	<i>Packet Data Structure, Header SerialTag Name</i>	<i>Header SerialTag Value</i>	<i>Information Description</i>
OUT	Calibration_PacketType CALIBRATION_SerialTag	4	Info. For remote presentation of calibration points. Generated by: *Various stages of calibration.
OUT	EyeData_PacketType EYEDATA_29_SerialTag	10	Eye data
IN	ASCII string data COMMAND_SerialTag	11	An ASCII string that is sent to the Settings Command Line Parser
IN	ASCII string data DATAFILE_INSERT_SerialTag	12	Immediately inserts the packet string into the data
IN / OUT	Variable length timeStamp data PING_SerialTag	13	IN: immediately returns identical copy of the timeStamp data packet in a PONG packet. Generated during the serial send test pattern.
IN / OUT	Variable length timeStamp data PONG_PacketType	14	IN: reports the round trip time in EventHistory. Generated in response to a PING packet with an exact copy of the timeStamp data.
OUT	Quality code REJECT_PacketType	16	Bad data

IN Packet Types are received by *ViewPoint*

OUT Packet Types are generated by *ViewPoint*

13.8 Data Value Encoding

All spatial location values, for example, position of gaze, calibration point locations, etc., are normalized and multiplied by 10,000 before being sent as a two byte integer (i.e., an unsigned short) that we will denote as **UINT16**. Thus all locations are stored in a TenKCode with range of 0 to 10,000. The following constants will help :

```
#define Normalized_to_TenKCode 10000.0
#define TenKCode_to_Normalized ( 1.0 / Normalized2TenKCode)
```

For example, the position of gaze data is converted to TenKCode format like this:

```
EyeData_PacketType dat ;
dat.gx = (UINT16)( normalXpos.x * Normalized_to_TenKCode );
dat.gy = (UINT16)( normalXpos.y * Normalized_to_TenKCode );
...
```

To reclaim the normalized floating point location value, simply divide the floating point value by 10000.0 :

```
if (MotorolaMachine) {
    normalxpos.x = dat.gx * TenKCode_to_Normalized;
    normalypos.y = dat.gy * TenKCode_to_Normalized;
}
elseif (IntelMachine) {
    normalxpos.x = BigToLittleEndian (dat.gx) * TenKCode_to_Normalized;
    normalypos.y = BigToLittleEndian (dat.gy) * TenKCode_to_Normalized;
}
```

Finally, to display the position in a window, multiply the normalized values by the window width and height and cast to integers :

```
POINT windowXpos;
windowXpos.x = (int)( normalXpos.x * windowHeight );
windowXpos.y = (int)( normalXpos.y * windowWidth );
```

13.9 Packet Data Structures

```
typedef enum
{
    HEAD_CalEpoch,      // Never sent
    START_CalEpoch,    // open full screen calibration window
    SHOW_CalEpoch,     // show calibration stimulus point
    CLICK_CalEpoch,    // show that point is registered, e.g. change color
    HIDE_CalEpoch,     // hide calibration point
    STOP_CalEpoch,     // close full screen calibration window
    TAIL_CalEpoch      // Never sent
} CalibrationEpochType ;

typedef struct          // CALIBRATION_SerialTag == 4
{
    UINT16 epochCode;  // type cast with (CalibrationEpochType)
    UINT16 pointIndex; // calibration point index
    UINT16 cx;         // X location of calibration point sent as TenKCode
    UINT16 cy;         // Y location of calibration point sent as TenKCode
} Calibration_PacketType;

typedef struct          // EYEDATA_29_SerialTag == 10
{
    UINT16 gx;         // X position sent as TenKCode
    UINT16 gy;         // Y position sent as TenKCode
    UINT16 dt;         // delta time in milliseconds since last eye data packet
    UINT16 bx;         // window discriminator box code
    UINT16 px;         // pupil width (normalized wrt EyeCamera window) sent as TenKCode
    UINT16 py;         // pupil height( normalized wrt EyeCamera window) sent as TenKCode
    UINT16 ct;         // cyclotorsion in minutes (degrees * 60)
    UINT16 ik;         // index of eye data packet
} EyeData_PacketType;
```

13.10 Example Serial Port Code

```
#define DataFile_InsertMarker_SerialTag 2
#define EventHistoryString_SerialTag 3
#define CommandLine_SerialTag 11
#define DataFile_InsertString_SerialTag 12

/*-----
/ serialSend_Command
/-----*/
void serialSend_CommandLine ( CSTR cstr )
{
    if ( haveSerialPort )
        serialOutputDataBytes( CommandLine_SerialTag, cstr, strlen(cstr) );
}

/*-----
/ serialSend_DataFile_InsertString
/-----*/
void serialSend_DataFile_Insert ( CSTR cstr )
{
    if ( haveSerialPort )
        serialOutputDataBytes( DataFile_InsertString_SerialTag, cstr, strlen(cstr) );
}
```

```

/*-----
/ serialOutputDataBytes
/-----*/

#define MaxPacketBytes 256
#define HeaderBytes 4
#define MaxBodyBytes  MaxPacketBytes - MaxHeaderBytes

void serialOutputDataBytes( SerialCodeType packetType, char data[], int packetBytes )
{
    char buffer[ MaxPacketBytes ] = { 'V', 'P', '0', '0' };
    char *body = buffer + HeaderBytes; // point past the packet header info
    buffer[0] = 'V' ;
    buffer[1] = 'P' ;
    buffer[2] = (char)( packetType );
    buffer[3] = (char)( packetBytes );
    memcpy( body, data, packetBytes );
    write( serialPort, buffer, packetBytes+ MaxHeaderBytes);
    flush( serialPort );
}

/*-----
/ SerialSend_DataFile_InsertMarker( 'K' );
/-----*/
SerialSend_DataFile_InsertMarker( char theMarkerChar )
{
    BYTE buffer [ 5 ] ;
    BYTE numberOfBytesInThePacketBody = 1 ;
    // Construct PACKET HEADER
    buffer[0] = (BYTE)( 'V' ) ; // the ascii capital character V
    buffer[1] = (BYTE)( 'P' ) ; // the ascii capital character P
    buffer[2] = DataFile_InsertMarker_SerialTag ; // =2
    buffer[3] = numberOfBytesInThePacketBody ;
    // Construct PACKET BODY
    buffer[4] = (BYTE)( theMarkerChar ) ;
    // Send packet.
    write( serialPort, &buffer, sizeof(buffer) );
    flush( serialPort );
}

serialSend_CommandLine ( "velocityThreshold 0.8" );
serialSend_CommandLine ( "dataFile_NewUnique" );
serialSend_DataFile_InsertString ( "This is an insert string from remote computer " );
serialSend_DataFile_InsertString ( "Remote tab data:\t0.0\t9.1\t8.2\t7.3" );
serialSend_CommandLine ( "dataFile_Pause" )

```

Hints:

Interfaces from some programs seem to require adding extra white spaces at the end of strings. Also make sure that the strings are zero terminated, rather than newline terminated. You can check the contents of incoming packets by selecting:

GUI menu item: Interface > Serial Port > Verbose Receive
 CLP command: verbose +serialReceive // +serialSend

Chapter 14 ViewPoint Interface: GUI, SDK, CLP

14.1 General

This Chapter provides a description of each GUI controls and the equivalent CLP commands and SDK functions.

There are several ways to interact with the *ViewPoint EyeTracker*[®]. The most apparent way is by using the graphical user interface (GUI) that consists of menu items, sliders, buttons, etc. Each of the GUI controls has a corresponding command line parser (CLP) string. The GUI control values can be saved in a Settings file that consists of the CLP strings (keyTerms and parameters), so control values can be easily loaded at any time. The CLP strings can also be sent from other programs, either on the same machine, or from another computer. The software developer's kit (SDK) includes a routine that allows other programs to send the CLP strings as well as providing access to data and other information.

14.1.1 VPX_SendCommand("setSomething") replaces VPX_SetSomething

Previous versions of ViewPoint provided individual SDK functions to set controls (VPX_Set* commands), such that there were equivalent GUI, SDK, and CLP instructions. With the introduction of the SDK function VPX_SendCommand(string), the need for equivalent SDK functions was largely obviated. In general, no new SDK VPX_Set* functions are being added and the old ones are considered an unnecessary duplication and are therefore being deprecated. **Future versions of ViewPoint may not support the VPX_Set* functions if there is an equivalent CLP command string that can be sent.** The deprecated SDK functions are printed in light gray.

14.1.2 VPX_SendCommand & formatted strings

The VPX_SendCommand function includes a va_list mechanism to handle formatted text. This means that it accept both simple string arguments and also strings with formatting instructions followed by additional arguments, just like the C language printf and scanf functions. This is extremely convenient for C programming, however, some third party applications (such as Matlab) do not provide an interface that handles va_lists. To accommodate these we include the VPX_SendCommandString function that takes only simple string arguments.

With formatted strings we can simply write:

```
VPX_SendCommand( "stimulus_BackgroundColor %d %d %d", r, g, b ); // formatted
```

Without formatted strings we must write:

```
char str[255];  
sprintf( str, "stimulus_BackgroundColor %d %d %d", r, g, b );  
VPX_SendCommandString( str ); // formatted
```

The programmer should use a precompiler conditional found inside the header file VPX.h.

```
#ifndef _IMPORTING_INTO_MATLAB
#define VPX_SendCommand VPX_SendCommandString
#else
VPX_DECLARE int VPX_SendCommand( TCHAR * szFormat, ...);
#endif
```

14.1.3 Quoting strings with white spaces

File names that contain spaces (e.g.: "My New File .txt") must be put inside double quotes. In a settings file this is very straight forward and readable:

```
dataFile_NewName " C:\VP\Data Files\Exp 6\subj 2.wks\"
```

However when specified withing a formatted string, the programmer must remember to escape the quote characters inside the command string, that is, precede the quote character with a back slash (e.g.: \"), as seen here:

```
VPX_SendCommand( "dataFile_NewName \"%s\" ", dataFileName );
VPX_SendCommand( "dataFile_InsertString \" Showing picture of a cat. \" ");
```

File names cannot begin with a white space. All white spaces are gobbled up until either the beginning of a string is specified by either a non-white space character, or a beginning quote is encountered.

14.1.4 Case insensitive CLP strings

The CLP command strings are generally presented starting with lower case and then capitalizing the first letter of successive words, however the parser does not care about the case of the command strings, so you do not need to worry about this as a source of error.

14.1.5 Boolean Toggle

As of version 2.8.2 all CLP commands that accept BoolValue arguments (e.g.: True, False, On, Off) can now also accept the argument "Toggle", e.g., "dataFile_Pause Toggle", that changes from the current state to the opposite state.

14.1.6 SDK return values

All SDK functions return the integer value 1 unless otherwise specified. Check the SDK header file, VPX.h, for final authority; changes may appear there before the documentation can be updated.

14.2 Data Files

14.2.1 Open Data File with Randomly Generated Name		
GUI:	File > Data > Unique Data File	^U
CLP Command:	dataFile_NewUnique	
SDK Function:	VPX_DataFile_NewUnique();	
<p>Opens a new data file with a unique new name without going through the "Save As" window. The data line in the Status window shows the data file name (default is a random number).</p> <p>By default data files are stored in the "Data" folder that is located in same folder as the ViewPoint application program. However this default can be overridden with the setPath command.</p> <p>See also:</p> <ul style="list-style-type: none"> setPath dataFile_NewUniqueExtension <p>See 10.3 Data File Format for a description of the format of the data file.</p>		
<pre>VPX_SendCommand("dataFile_NewUnique"); VPX_DataFile_NewUnique();</pre>		

14.2.2 Specify NewUnique Data File Extension		
GUI:	-none	
CLP Command:	dataFile_NewUniqueExtension <i>fileExtensionString</i>	
SDK Function:	-none-	
Default:	.txt	
<p>Specifies the file type extension that is appended when performing the dataFile_NewUnique operation. This specification does <i>not</i> affect the format of the data inside the file. This specification does <i>not</i> affect the dataFile_NewName command.</p> <p>A dot should be included for this to correctly specify a file type, e.g. ".wks", ".txt", ".doc". The file extension ".wks", will usually cause the data file to be opened directly by <i>Microsoft Excel</i> or <i>Microsoft Works</i> spreadsheet packages.</p> <p>See section: 14.1.3 Quoting strings with white spaces</p> <p>HINT: You can use the extension string to append a group name, e.g., "_drug_C_.wks" or "_MondayData.txt",</p> <p>v.2.8.1.12 CLP added.</p>		
<pre>dataFile_NewUniqueExtension " drug group X .xls" VPX_SendCommand("dataFile_NewUniqueExtension .txt "); VPX_SendCommand("dataFile_NewUniqueExtension \" patient_%d .txt\" ", pid);</pre>		

14.2.3 Open a Data File and Specify a File Name

GUI: **File > Data > New Data File ...** **^N**
 CLP Command: **dataFile_NewName** *fileNameString*
 SDK Function: **-none-**

Opens a new data file with a specified name. The GUI Menu selection allows the user to specify the file name through the "New ViewPoint Data File" dialog.

Any file type extension must be included in the string.

The `dataFile_NewUniqueExtension` specification *does not* affect this command.

See section: 14.1.3 Quoting strings with white spaces

The Data line in the Status window reports the current state of data recording.

```
VPX_SendCommand( "dataFile_NewName myData.txt" );
VPX_SendCommand( "dataFile_NewName \" C:\\VP\\Data Files\\Exp 6\\subj 2.wks\\" );
VPX_SendCommand( "dataFile_NewName \"%s\" ", dataFileName );
```

14.2.4 Insert a String into the Data File

GUI: **-none-**
 CLP Command: **dataFile_InsertString** *string*
 SDK Function: **-none-**

Inserts the string into the data file. The string must be inside quotes if it contains white spaces; in C programs the quote characters inside the command string must be escaped with backslashes.

See section: 14.1.3 Quoting strings with white spaces

The string can either inserted synchronously, i.e., at the end of the next data line (record), or asynchronously, i.e., on a separate line, depending upon the specification of **dataFile_AsynchStringData**

When called more frequently than data is saved, and the **dataFile_AsynchStringData** is set to false, then the strings are concatenated. The string separator is the tab character.

See also:

dataFile_InsertMarker
dataFile_AsynchStringData

```
VPX_SendCommand( "dataFile_InsertString showingPictureOfCat" );
VPX_SendCommand( "dataFile_InsertString \" Showing picture of a cat. \" ");
VPX_SendCommand( "dataFile_InsertString \"%s\" ", userString );
```

14.2.5 Insert a Marker into the Data File

GUI: **-none-**

CLP Command: **dataFile_InsertMarker DataMarker**
DataMarker: Any single byte ascii character.

SDK Function: **VPX_DataFile_InsertMarker (DataMarker);**

Insert the specified ASCII character into the data file. This can be used for data synchronization coding, or patient responses, etc.

The string can either inserted synchronously, i.e., at the end of the next data line (record), or asynchronously, i.e., on a separate line, depending upon the specification of **dataFile_AsynchMarkerData Yes**

Where exact timing of the markers is critical, you may achieve better performance using the specific SDK function directly, rather than sending the CLP command string that must first be processed by the parser.

Try the program **~/ViewPoint/ExtraApps/DataMarker.exe** provides an easy way to manually insert markers into the data file in real-time.

Note:

Currently, non-printable ascii characters are not filtered out, so be careful if you specify such characters as: tab, bell, backspace, line-feed, etc.

See also:

dataFile_InsertString

dataFile_AsynchMarkerData

```
dataFile_InsertMarker K
VPX_SendCommand ("dataFile_InsertMarker K ")
VPX_SendCommand ("dataFile_InsertMarker %c ", theMarker )
VPX_DataFile_InsertMarker ('K')
```

14.2.6 Insert a User Defined Data Tag into the Data File	
GUI:	-none-
CLP Command:	dataFile_InsertUserTag <i>UserTag</i>
SDK Function:	-none-
<p>This allows users to insert uniquely tagged data from their own sources into a data file. The insertion is done asynchronously with respect to the eye movement data records and the insertions are uniquely time stamped.</p> <p>See: Section 10.3.2 on page 52.</p> <p>See also:</p> <p>dataFile_InsertString</p>	
<pre>dataFile_InsertUserTag 800 "MyUserData 1 A 0.888" VPX_SendCommand("dataFile_InsertUserTag 800 \"MyUserData\t%d\t%d\" ", ix, ix*ix)</pre>	

14.2.7 Specifies asynchronous or synchronous string data	
GUI:	File > Data > Asynchronous String Data
CLP Command:	dataFile_AsynchStringData <i>BoolValue</i> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
Default:	Yes
<p>Specifies whether to insert strings data asynchronously or synchronously into the data file. Synchronously means that this data is appended to the same line as the normal eye tracker data. This string data will be treated as multi-column data by using tab-characters as column separators. Synchronous data is usually easier to load into spread sheets or other analysis packages. If several Strings are inserted between eye tracker samples, the Strings will all be concatenated. If this control is set to Yes (i.e., Asynchronous), then each String is separately time stamped and inserted on a data line by itself.</p> <p>See: Section 10.3.2 on page 52.</p> <p>See also:</p> <p>dataFile_InsertString</p>	
<pre>VPX_SendCommand("datafile_ AsynchStringData No");</pre>	

14.2.8 Specify asynchronous or synchronous markers data	
GUI:	File > Data > Asynchronous Marker Data
CLP Command:	<code>dataFile_AsynchMarkerData BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
Default Setting:	No
<p>Specifies whether to insert data markers asynchronously or synchronously into the data file. Synchronously means that this data is on the same line as the normal eye tracker data, in a separate column, which is usually easier to load into spread sheets or other analysis packages. If several Markers are inserted between eye tracker samples, the Markers will all be displayed together and more precise Marker time information is lost. If this control is set to Yes (i.e., Asynchronous), then each Marker event is separately time stamped and inserted on a data line by itself.</p> <p>See: Section 10.3.2 on page 52.</p> <p>See also: dataFile_InsertMarker</p>	
<pre>VPX_SendCommand("datafile_AsynchMarkerData Yes");</pre>	

14.2.9 Specify asynchronous or synchronous head tracker data	
GUI:	File > Data > Asynchronous Head Tracker Data
CLP Command:	<code>dataFile_AsynchHeadData BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
Default:	Yes
<p>Available only with head tracker option</p> <p>Specifies whether to insert head tracker data asynchronously or synchronously into the data file.</p> <p>See: Section 10.3.2 on page 52.</p> <p>See also: headTrackerConnect</p>	
<pre>VPX_SendCommand("dataFile_AsynchHeadData yes");</pre>	

14.2.10 Specify data file start time	
GUI:	File > Data > Start Data File Time at Zero
CLP Command:	<code>dataFile_startFileTimeAtZero BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
Default Setting:	Yes
<p>Specifies whether to start each new data file at time=0. Otherwise, the data-file will use the time from the DLL, so the time values in each sequential data file will be increasing and represent actual elapsed time. The DLL time starts at zero when the DLL is launched, which is when the first program that accesses it is launched. Turning this off may be useful to keep track of fatigue factors or the duration of rest periods. It is may also be useful because other programs that use the DLL time will have the same time values, which can aid in post-hoc synchronization of events.</p> <p>See also: VPX_GetPrecisionDeltaTime</p>	
<pre>VPX_SendCommand("datafile_StartFileTimeAtzero No");</pre>	

14.2.11 Store smoothed or unsmoothed data	
GUI:	File > Data > Store Smoothed Data
CLP Command:	<code>dataFile_StoreSmoothedData BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	<code>-none-</code>
Default:	No
<p>Specifies whether to store the trailing average smoothed data using the number of smoothing points specified by the Smoothing Points Slider on the Controls window. The eye trace lines displayed in the GazeSpace and Stimulus windows can be smoothed to reduce noise. The degree of smoothing of gaze calculation may be varied using the slider on the Controls window (see figure 9). When placed at the far left, no smoothing is performed. Incrementing the slider to the right increases the number of previous points included in the average calculation. A value of 4 makes attractive and useful real-time graphics. Smoothing effects the real time calculations. Because smoothing effects the velocity calculations the saccade velocity threshold must be adjusted proportionately. Smoothing will also affect which ROI boxes are triggered. By default, smoothing does NOT influence the data values that are stored to file, because the real-time smoothing uses a trailing average technique, whereas post hoc data analysis should use a symmetrical smoothing technique.</p> <p><i>See also:</i> <code>smoothingPoints, smoothingMethod</code></p>	
<pre>VPX_SendCommand("datafile_StoreSmoothedData Yes");</pre>	

14.2.12 Specify whether to use buffering (DEPRECATED)	
GUI:	<code>-removed-</code>
CLP Command:	<code>dataFile_UseBuffering BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	<code>VPX_DataFile_Buffering(bool tf);</code>
Default:	Yes
<p>DEPRECATED</p> <p>Allows selection between (Yes) buffer the data in RAM before saving to disk, or (NO) immediately write to disk each data record. Because previous versions MSWindows were quite unstable, many users preferred to turn buffering off, so that data was not lost in the event of a system or program crash; this did however sometimes incur a slower sampling rate. In general this is neither required nor recommended with most newer operating systems.</p>	
<pre>VPX_SendCommand("dataFile_UseBuffering No"); VPX_DataFile_Buffering(false);</pre>	

14.2.13 Pause writing of data to file		
GUI:	File > Data > Pause Data Capture (toggle)	^P
CLP Command:	dataFile_Pause dataFile_Resume <i>No arguments.</i>	
SDK Function:	VPX_DataFile_Pause(bool tf);	
<p>Pauses the writing of data to an open data file. Inserts a "+" marker into the data file when paused and inserts a "=" marker at time resumed. Because of the MSWindows overhead for opening and closing files, the user may prefer pausing and resuming to opening and closing. Also, pause may be set before the file is opened, such that the overhead delays for opening the file are finished before the start of the experiment. The Data line in the Status window reports the current state of data recording.</p>		
<pre>VPX_SendCommand("dataFile_Pause"); VPX_SendCommand("dataFile_Resume"); VPX_DataFile_Pause(TRUE);</pre>		

14.2.14 Close Data File		
GUI:	File > Data > Close Data File	^W
CLP Command:	dataFile_Close	
SDK Function:	VPX_DataFile_Close();	
<p>Closes a data file if one is open, regardless of Pause state. The Data line in the Status window reports the current state of data recording. See also: dataFile_CloseAndAnalyze</p>		
<pre>VPX_SendCommand("dataFile_Close"); VPX_DataFile_Close();</pre>		

14.2.15 Close Data File and Open in Post-Hoc Analysis tool		
GUI:	File > Data > Close & Analyze Data File	Alt-Shift-W
CLP Command:	dataFile_CloseAndAnalyze	
SDK Function:	-none-	
<p>Closes a data file if one is open, regardless of Pause state, and automatically launches the ViewPoint DataAnalysis™ program with the data file loaded in.</p> <p>The Data line in the Status window reports the current state of data recording.</p>		
<pre>VPX_SendCommand("dataFile_CloseAndAnalyze"); VPX_closeAndAnalyzeDataFile();</pre>		

14.3 Stimulus Images

14.3.1 Load Stimulus Image into the Stimulus window		
GUI:	File > Images > Load Image	^I
CLP Command:	stimulus_LoadImageFile <i>filename</i>	
SDK Function:	-none-	
<p>Loads the selected image into the Stimulus window</p> <p>See section: 14.1.3 Quoting strings with white spaces</p> <p>See also:</p> <p>stimulusGraphicsOptions +Image</p>		
<pre>VPX_SendCommand("stimulus_LoadImageFile catPicture.bmp"); VPX_SendCommand("stimulus_LoadImageFile \"second cat picture .bmp\" "); VPX_SendCommand("stimulus_LoadImageFile %s", bitmapFileName);</pre>		

14.3.2 Specifies how to display the currently loaded stimulus image	
GUI	Stimuli > Image Shape > shape type
CLP Command:	stimulus_ImageShape <i>ShapeType</i> <i>ShapeType: Actual, Centered, Fit, Isotropic</i>
SDK Function:	-none-
Default:	Fit
<p>Specifies how the bitmap image is to be displayed in the Stimulus windows.</p> <p>A = Actual : Displays the in the two windows at actual size.</p> <p>C = Centered : Displays the image actual size and centered in the windows.</p> <p>F = Fit : Displays the image stretched un-equally to fit the window.</p> <p>I = Isotropic: Displays the image stretched equally in all directions, so as to maintain the original proportions, i.e., the original aspect ratio of the image. This may leave window background ("matting") color between edges of the picture and the edges of the window. The color of this area can be specified with the command: stimulus_BackgroundColor. Note: the gazeSpace window may automatically resize to accommodate.</p> <p>See also:</p> <p>stimulus_BackgroundColor</p> <p>VPX_STATUS_StimulusImageShape</p>	
<pre>VPX_SendCommand("stimulus_ImageShape Isotropic");</pre>	

14.3.3 Specify a background "matting" color for the stimulus window	
GUI:	Stimuli > Background Color
CLP Command:	stimulus_BackgroundColor <i>ColorValue</i> <i>ColorValue: Red 0-255, Green 0-255, Blue 0-255</i>
SDK Function:	-none-
<p>Added the ability to set the background ("matting") color for use when ImageShape is not set to Fit and there is space at the sides or at the bottom of the image.</p> <p><i>Workarounds:</i> The window may need to be minimized and reopened to show the color changes.</p> <p>See also:</p> <p>stimulus_ImageShape</p>	
<pre>Eg.set to BRIGHT RED use: stimulus_BackgroundColor 255 0 0 VPX_sendCommand ("stimulus_BackgroundColor 255 135 075");</pre>	

14.3.4 Play specified Sound file	
GUI:	-none-
CLP Command:	stimulus_PlaySoundFile <i>soundFileName</i>
SDK Function:	-none-
<p>Plays the specified sound file. May be used as an auditory cue. If the string contains spaces it must be in quotes. e.g stimulus_PlaySoundFile "Yes.wav".</p> <p>See section: 14.1.3 Quoting strings with white spaces</p> <p>This feature may cause a media/audio player to open, if this happens, check your computer settings.</p>	
<pre>stimulus_PlaySoundFile "a very loud meow .wav" VPX_SendCommand("stimulus_PlaySoundFile meow.wav"); VPX_SendCommand("stimulus_PlaySoundFile \"a very loud meow .wav\" "); VPX_SendCommand("stimulus_PlaySoundFile \"%s\" ", soundFileName);</pre>	

14.4 PictureList

14.4.1 Initialize Picture List	
GUI:	-none-
CLP Command:	pictureList_Init
SDK Function:	-none-
<p>Initializes the list for stimulus images, making it ready for new names to be entered.</p>	
<pre>pictureList_Init VPX_SendCommand("pictureList_Init");</pre>	

14.4.2 Add List of Image Names to PictureList	
GUI:	-none-
CLP Command:	pictureList_AddName <i>imageFileName</i>
SDK Function:	-none-
<p>Adds an image file name to the picture list.</p>	
<pre>pictureList_AddName "picture of cat .bmp" VPX_SendCommand ("pictureList_AddName picture2.bmp"); VPX_SendCommand ("pictureList_AddName \"picture of cat .bmp\" "); VPX_SendCommand ("pictureList_AddName \"%s\" ", imageFileName);</pre>	

14.4.3 Randomize List of Images in the PictureList	
GUI:	File > Images > Picture List > Randomize PictureList
CLP Command:	<code>pictureList_Randomize</code>
SDK Function:	-none-
Randomizes the pointers in the picture list. Repeat this to re-randomize.	
<code>VPX_SendCommand ("pictureList_Randomize");</code>	

14.4.4 Move to Next Image in the PictureList	
GUI:	File > Images >PictureList>Next PictureList Image ^F12 (default)
CLP Command:	<code>pictureList_ShowNext</code>
SDK Function:	-none-
Moves to the next file pointer in the picture list.	
<code>VPX_SendCommand ("pictureList_ShowNext");</code>	

14.4.5 Move to Start of Images in Picture List	
GUI:	File > Images > PictureList > Restart PictureList
CLP Command:	<code>pictureList_Restart</code>
SDK Function:	-none-
Re-sets the pointer to the first image in the list. Does not un-randomize if the list has been randomized.	
<code>VPX_SendCommand ("pictureList_Restart");</code>	

14.5 Controls window: VideoImage

14.5.1 Specify Mapping Feature	
GUI:	Controls Window, Video Image tab, Mapping feature pull down menu
CLP Command:	<code>mappingFeature <i>Method</i></code> <i>Methods:</i> Pupil, Glint, Vector, Manual, SlipComp
SDK Function:	<code>int VPX_SetFeatureMethod (int <i>featureMethod</i>)</code> <code>int VPX_SetFeatureMethod2(VPX_EyeType eyn, int <i>featureMethod</i>)</code> <i>featureMethod :</i> <code>PUPIL_ONLY_Method, GLINT_ONLY_Method, VECTOR_DIF_Method</code>
Default:	Pupil
Controls what type of mapping will be done from EyeSpace to GazeSpace	
Return value: 0=false, 1=true, otherwise -1 if invalid value for eye	
<pre>VPX_SendCommand ("mappingFeature Glint"); VPX_SendCommand ("mappingFeature Vector"); VPX_SendCommand ("mappingFeature Pupil"); VPX_SendCommand ("mappingFeature Manual"); VPX_SendCommand ("mappingFeature SlipComp");</pre>	

14.5.2 AutoThreshold	
GUI:	Controls window, VideoImage tab, Threshold: Autothreshold button
CLP Command:	<code>autoThreshold</code> <code>autoThreshold_Start</code>
SDK Function:	<code>int VPX_AutoThreshold();</code> <code>int VPX_AutoThreshold2(VPX_EyeType eye);</code> <i>VPX_EyeType : Eye_A, Eye_B</i>
Automatically sets desirable glint and pupil threshold levels.	
<pre>VPX_SendCommand ("autoThreshold"); VPX_AutoThreshold2(EYE_A);</pre>	

14.5.3 Positive Lock Tracking	
GUI:	Controls Window, Video Image Tab, Threshold Group, Positive-Lock Threshold-Tracking
CLP Command:	<code>positiveLock BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	<code>int set_positiveLockThresholdTracking(int boolValue);</code> <code>int VPX_SetPositiveLockThresholdTracking2(VPX_EyeType, int tf)</code> Return value: 0=false, 1=true, otherwise -1 if invalid value for eye
Default:	<code>off</code>
Continuous automatic feature threshold adjustment.	
<code>VPX_SendCommand ("positiveLock on");</code>	

14.5.4 Adjust Pupil Threshold Slider	
GUI:	Controls window, VideoImage tab, Pupil / Threshold slider
CLP Command:	<code>pupilThreshold NormalizedValue</code> <code>pupilThreshold EyeType NormalizedValue</code> <i>NormalizedValue: a floating point number in range 0.0 to 1.0</i> <i>EyeType: Eye_A, Eye_B</i>
SDK Function:	<code>int VPX_SetPupilThreshold (EyeType normalizedValue)</code> <code>int VPX_SetPupilThreshold2(VPX_EyeType eyn, float value)</code> Return value: 0=false, 1=true, otherwise -1 if invalid value for eye
Default:	<code>0.25</code> , but autothreshold at startup may reset this
Sets the image intensity threshold such that the pupil can be segmented from the rest of the image. The assumption is that the pupil is darker than the rest of the image within the PupilScanArea. For binocular systems, the user may specify this value separately for eye. The value zero represents black, the lowest pixel intensity possible, and the value one represents white, the highest pixel intensity possible. <i>See also:</i> <code>autoThreshold</code>	
<code>VPX_SendCommand ("pupilThreshold 0.7");</code>	

14.5.5 Adjust Glint Threshold Slider	
GUI:	Controls window, VideoImage tab, Glint / Threshold slider
CLP Command:	<code>glintThreshold <i>NormalizedValue</i></code> <i>NormalizedValue</i> : a floating point number in range 0.0 to 1.0
SDK Function:	<code>int VPX_SetGlintThreshold (float <i>normalizedValue</i>);</code> <code>int VPX_SetGlintThreshold2(VPX_EyeType <i>eyn</i>, float <i>value</i>);</code>
Default:	0.88, but autothreshold at startup may reset this
<p>Sets the level pixel intensity used for segmenting the image when searching for the glint (aka, corneal reflection, corneal reflex, or 1st Purkinje image). All pixels with intensity greater than this value are candidates for being classified as the glint.</p> <p>The value zero represents black, the lowest pixel intensity possible, and the value one represents white, the highest pixel intensity possible.</p> <p>See also: autoThreshold</p>	
<code>VPX_SendCommand ("glintThreshold 0.6");</code>	

14.5.6 Adjust Video Image Brightness	
GUI:	Controls window, VideoImage tab, Brightness slider
CLP Command:	<code>videoImageBrightness <i>NormalizedValue</i></code> <i>NormalizedValue</i> : a floating point number in range 0.0 to 1.0
SDK Function:	<code>int VPX_SetImageBrightness (float <i>normalValue</i>)</code> <code>int VPX_SetImageBrightness2(VPX_EyeType <i>eyn</i>, float <i>normalVal</i>)</code> <i>normalValue</i> : a floating point number in range 0.0 to 1.0
Default:	- varies -
Sets the video image brightness levels normalized from 0.0 to 1.0.	
<code>VPX_SendCommand("videoImageBrightness 0.5");</code>	

14.5.7 Adjust Video Image Contrast	
GUI:	Controls window, VideoImage tab, Contrast slider
CLP Command:	<code>videoImageContrast <i>normalValue</i></code> <i>normalValue</i> : a floating point number in range 0.0 to 1.0
SDK Function:	<code>int VPX_SetImageContrast (float <i>normalValue</i>);</code> <code>int VPX_SetImageContrast2(VPX_EyeType eyn, float <i>normalVal</i>);</code> <i>normalValue</i> : a floating point number in range 0.0 to 1.0
Default:	- varies -
Sets the video image contrast levels normalized from 0.0 to 1.0.	
<code>VPX_SendCommand("videoImageContrast 0.7");</code>	

14.5.8 Dynamically Optimize Brightness and Contrast Settings	
GUI:	Controls window, VideoImage tab, AutoImage checkbox
CLP Command:	<code>videoAutoImage <i>BoolValue</i></code>
SDK Function:	-none-
Default:	Off
Continuously attempts to set the video image brightness and contrast levels to optimal values. <i>Note 1</i> : Only the region within the pupil scan area is examined, the pupil scan area rectangle must be of sufficient size for the algorithm to sample a range of gray levels, otherwise the algorithm will fail. <i>Note 2</i> : The algorithm is under development and may change without notice.	
<code>VPX_SendCommand("videoAutoImage On");</code>	

14.5.9 Adjust Pupil Scan Density	
GUI:	Controls window, Videolmage tab, Pupil / Scan Density slider
CLP Command:	<code>pupilScanDensity <i>FloatValue</i></code> <i>FloatValue</i> : integer in range 1 to 20
SDK Function:	<code>VPX_SetPupilResolution (float <i>resolution</i>);</code> <code>VPX_SetPupilResolution2 (<i>VPX_EyeType</i> eye, float <i>resolution</i>);</code> <i>VPX_EyeType</i> : Eye_A, Eye_B <i>resolution</i> : use only whole value in range 1 to 20
Default:	7
<p>The value specifies the pixel sampling interval for the threshold segmentation operation. The value 1 indicates to sample every pixel. The value 2 indicates to sample every other pixel in both x and y directions, so one fourth as many pixels are sampled with a setting of 2 as with a setting of 1. Etc.</p> <p>Caution: normally there is no need to sample very densely and doing so will greatly burden the cpu. For the GUI interface the slider has a default minimal value greater than 1 to avoid cpu overload. This default minimum may be change with the CLP command: <code>minimumPupilScanDensity.</code></p> <p>Note: previous versions provided for either a normalized floating point value in the range (0.0 – 1.0), or an integer in the range 1 – 20, however the normalized floating point values are no longer supported. There is no confusion with the value 1, because the minimal sampling interval of integer 1 and the maximum normalized density (1.0) are opposite ways of looking at the same thing.</p> <p>See also: <code>minimumPupilScanDensity</code></p>	
<code>VPX_SendCommand ("pupilScanDensity 5");</code>	

14.5.10 Override Pupil Scan Density Minimum	
GUI:	-none-
CLP Command:	<code>minimumPupilScanDensity <i>DensityIndex</i></code> <i>DensityIndex: Integer in range 3 to 20, depending upon max density chosen.</i>
SDK Function:	-none-
Default:	7
<p>Overrides the minimum pupil scan density on the Controls window slider. Fine sampling is rarely required and not generally recommended. WARNING: Setting the scan density too fine can create a huge burden on the CPU and possibly lock-out use of the GUI.</p>	
<pre>VPX_SendCommand ("minimumPupilScanDensity 12");</pre>	

14.5.11 Adjust Glint Scan Density	
GUI:	Controls window, Videolmage tab, Glint / Scan Density slider
CLP Command:	<code>glintScanDensity IntValue</code> <i>IntValue</i> : integer in range 1 to 20
SDK Function:	<code>VPX_SetGlintResolution (float resolution);</code> <code>VPX_SetGlintResolution2 (VPX_EyeType eye, float resolution);</code> <i>VPX_EyeType</i> : Eye_A, Eye_B <i>resolution</i> : use only whole value in range 1 to 20
Default:	2 or 3 , depending on glintSegmentationMethod
<p>The argument specifies the pixel sampling interval for the glint threshold segmentation operation. The value 1 indicates to smple every pixel. The value 2 indicates to sample every other pixel in both the x and y directions, so one fourth as many pixels are sampled as with a setting of 1. Etc. The glint is usually much smaller than the pupil, so finer sampling is expected.</p> <p>Caution: normally there is no need to sample very densely and doing so will greatly burden the cpu. For the GUI interface the slider has a default minimal value greater than 1 to avoid cpu overload. This default minimum may be change with the CLP command: minimumGlintScanDensity.</p> <p>Note: previous versions provided for either a normalized floating point value in the range (0.0 – 1.0), or an integer in the range 1 – 20, however the normalized floating point values are no longer supported. There is no confusion with the value 1, because the minimal sampling interval of integer 1 and the maximum normalized density (1.0) are opposite ways of looking at the same thing.</p> <p>See also: <code>minimumGlintScanDensity</code> <code>glintSegmentationMethod</code></p>	
<code>VPX_SendCommand ("glintScanDensity 5");</code>	

14.5.12 Override Glint Scan Density Minimum	
GUI:	-none-
CLP Command:	<code>minimumGlintScanDensity <i>DensityIndex</i></code> <i>DensityIndex: Integer in range 7 to 20, depending upon max density chosen.</i>
SDK Function:	-none-
Default:	1 or 3, depending on glintSegmentationMethod
<p>Overrides the minimum glint scan density available on the Controls window slider. <i>Note:</i> this is not normally required and small scan density values can over burden the cpu. <i>See also:</i> glintScanDensity glintSegmentationMethod</p>	
<pre>VPX_SendCommand ("minimumGlintScanDensity 3");</pre>	

14.6 EyeCamera Window

14.6.1 Adjust Pupil Sacn Area	
GUI:	EyeCamera window , EyeCamera toolbar , Top Button
CLP Command:	<code>pupilScanArea <i>L T R B</i></code> <i>L T R B: Normalized floating point values for the corners.</i>
SDK Function:	<code>VPX_SetPupilScanArea (VPX_RealRect rr);</code>
Default:	0.200 0.200 0.800 0.800
<p>Defines scan area for the pupil. The four values are the floating point coordinates (0.0 – 1.0) of the bounding rectangle, listed in the order: Left, Top, Right, Bottom.</p>	
<pre>VPX_SendCommand ("pupilScanArea 0.3 0.2 1.0 0.4");</pre>	

14.6.2 Specify Pupil Scan Area Shape	
GUI:	-none-
CLP Command:	pupilScanShape <i>ScanShapeType</i> <i>ScanShapeType:</i> Rectangle, Ellipse.
SDK Function:	-none-
Default:	Elliptical
Specifies whether to change the scan area for the pupil to either rectangular or elliptical . Elliptical scan area is effective at eliminating dark spots that the software interprets as a pupil. Elliptical Scan	
<code>VPX_SendCommand("pupilScanShape Ellipse");</code>	

14.6.3 Pupil and Glint oval fit constraints	
GUI:	-none-
CLP Command:	pupilConstrained <i>BoolValue</i> glintConstrained <i>BoolValue</i>
SDK Function:	-none-
Controls whether or not the pupil (glint) oval fit is allowed to extend beyond the scan area rectangle in the EyeSpace window.	
<code>VPX_SendCommand("pupilConstrained True");</code>	

14.6.4 Define Glint Scan Area	
GUI:	EyeCamera window , EyeCamera toolbar , Third Button to enable Drag mouse in window to define the glint scan area rectangle.
CLP Command:	glintScanSize <i>X Y</i>
SDK Function:	VPX_SetGlintScanSize (VPX_RealPoint rp);
Default:	0.400 0.200
Defines scan area for the glint. The two values are the normalized floating point X,Y size values of the scan rectangle.	
<code>VPX_SendCommand ("glintScanSize -0.4 1.3");</code>	

14.6.5 Define Offset of Glint Sacn Area Relative to the Pupil	
GUI:	-none-
CLP Command:	<code>glintScanOffset X Y</code> <i>X Y : Normalized floating point coordinates.</i>
SDK Function:	<code>VPX_SetGlintScanOffset (VPX_RealPoint rp);</code>
Default:	0.010 0.080
<p>Defines offset of the glint scan area relative to the center of the pupil. The two values are the normalized coordinates of the offset vector from the center of the pupil.</p> <p>See also:</p> <p><code>video_yokedGlint NO</code> <code>glintScanUnYokedOffset</code></p>	
<code>VPX_SendCommand ("glintScanOffset 0.4 0.3");</code>	

14.6.6 Unyoke Glint Scan Area from the Pupil	
GUI:	-none-
CLP Command:	<code>video_yokedGlint BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
<p>Allows the glint scan area to be unyoked from the pupil. i.e. the glint scan area does not move with the pupil. Special applications only.</p> <p>See also:</p> <p><code>glintScanUnYokedOffset</code> <code>glintScanOffset</code></p>	
<code>VPX_SendCommand("video_yokedGlint On");</code>	

14.6.7 Define offset of Unyoked Glint Scan Area	
GUI:	-none-
CLPCommand:	<code>glintScanUnYokedOffset X Y</code> <i>X Y : Normalized floating point coordinates.</i>
SDK Function:	<code>VPX_SetGlintScanUnyokedOffset(VPX_RealPoint rp);</code>
<p>Defines offset of the glint scan area relative to the upper left hand corner of the EyeCamera window.</p> <p>The two values are the normalized coordinates of the offset vector.</p> <p>See also: video_yokedGlint</p>	
<code>VPX_SendCommand ("glintScanUnYokedOffset 0.1 -3.0");</code>	

14.6.8 Toggle Show Treshold Dots On / Off	
GUI:	Video > Show Threshold Dots (toggle) ^D Button on EyeCamera window
CLP Command:	<code>showThresholdDots BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	<code>VPX_Video_ShowThresholdDots (bool tf);</code>
Default:	Yes
<p>Specifies whether the image segmentation dots are displayed in the EyeCamera window.</p>	
<pre>VPX_SendCommand ("showThresholdDots No"); VPX_SendCommand ("showThresholdDots %s", boolVal?"Yes":"No");</pre>	

14.6.9 Specify EyeImage Overlay Graphics sent to layered application (EXPERIMENTAL)	
Control or Menu:	-none-
CLP Command:	<code>vpx_EyeCameraImageOverlays <i>StringArg</i></code> <code><i>StringArg</i> : +Eye_A -Eye_A +Eye_B -Eye_B</code>
SDK Function:	<code>VPX_SetEyeImageOverlays(VPX_EyeType eye, bool tf);</code> <code><i>VPX_EyeType</i> : Eye_A, Eye_B</code>
<p>EXPERIMENTAL : This functionality is under development and may not be stable over future versions.</p> <p>This only affects the overlay graphics in the remote eye image that is sent to a layered SDK application with the SDK function <code>VPX_SetEyeImageWindow</code>. It does not affect the EyeCamera window within the <i>ViewPoint</i>. Also, this does not effect the display of the segmentation dots; to remove these dots from both <i>ViewPoint</i> and the layered SDK application, use the command <code>showThresholdDots</code></p> <p>Note carefully: the CLP and the SDK names are spelled differently.</p> <p>See also:</p> <ul style="list-style-type: none"> <code>VPX_SetEyeImageWindow</code> <code>showThresholdDots</code> 	
<pre>VPX_SendCommand ("vpx_EyeCameraImageOverlays off"); VPX_SetEyeImageOverlays(Eye_A, 1);</pre>	

14.6.10 EyeCamera Tool Bar Display	
GUI:	Video > Show Toolbar on EyeCamera window
CLP Command:	<code>videoToolBar <i>BoolValue</i></code> <code><i>BoolValue</i>: Yes, No, True, False, On, Off, 1, 0, Toggle</code>
SDK Function:	<code>VPX_Video_ShowEyeCameraToolBar (bool tf);</code>
Default:	Yes
Specifies whether to display eye camera tool bar.	
<pre>VPX_SendCommand("videoToolBar off");</pre>	

14.7 Video related controls

14.7.1 Specify Video Input Standard	
GUI:	Video > Video Standard > NTSC, PAL, SECAM
CLP Command:	videoStandard <i>Option</i> <i>Option: any one of: NTSC, PAL, SECAM</i>
SDK Function:	-none-
Default:	NTSC upon first run, but saved in preferences file thereafter.
<p>Specifies which video mode to use.</p> <p>ViewPoint tries to ensure that it starts up in a friendly way each time, to facilitate this the videoStandard selection is saved in a special Preferences file that is evaluated each time it ViewPoint is launched.</p>	
<pre>VPX_SendCommand("videoStandard PAL"); VPX_SendCommand("videoStandard SECAM"); VPX_SendCommand("videoStandard NTSC");</pre>	

14.7.2 Specify Tracking Operation Mode	
GUI:	Video > Mode > Setup, Precision, Speed
CLP Command:	videoMode <i>ProcessingMode</i> <i>ProcessingMode: Setup, Precision, Speed</i>
SDK Function:	<code>VPX_Video_Mode (VideoProcessingMode mode);</code> <code>Setup_Mode, HighPrecision_Mode, HighSpeed_Mode</code>
Default:	Setup
<p>Specifies which operation mode to use.</p>	
<pre>VPX_SendCommand("videoMode Precision"); VPX_SendCommand("videoMode Speed"); VPX_SendCommand("videoMode Setup");</pre>	

14.7.3 Specify Dark or Bright Pupil Tracking	
GUI:	Video > Pupil Type > Dark Pupil, Bright Pupil
CLP Command:	<code>pupilType <i>PupilType</i></code> <i>PupilType:</i> Dark, Bright
SDK Function:	<code>int VPX_SetPupilType (PupilMethod);</code> <code>int VPX_SetPupilType2(VPX_EyeType eyn, int method);</code> <code>DARK_PUPIL_Method, BRIGHT_PUPIL_Method</code>
Specifies dark or bright pupil tracking.	
<code>VPX_SendCommand("pupilType dark");</code>	

14.7.4 Specify Pupil Segmentation Method	
GUI:	Video > Pupil Segmentation Method > Centroid, Oval Fit
CLP Command:	<code>pupilsegmentationMethod <i>Method</i></code> <i>Method:</i> Centroid, OvalFit
SDK Function:	-none-
Default:	OvalFit
Specifies which pupil segmentation method to use, either oval fit or centroid. Actually the centroid is obtained in either case, but OvalFit performs additional image processing and fitting. <i>Note:</i> the pupil width and pupil aspect ratio will not be available unless OvalFit is selected.	
<code>VPX_SendCommand("pupilsegmentationMethod Centroid");</code>	

14.7.5 Specify Glint Segmentation Method	
GUI:	Video > Glint Segmentation Method > Centroid, Oval Fit
CLP Command:	<code>glintsegmentationMethod <i>Method</i></code> <i>Method:</i> Centroid, OvalFit
SDK Function:	-none-
Default:	OvalFit
Specifies which glint segmentation method to use, either ovalFit or centroid. Actually the centroid is obtained in either case, but OvalFit performs additional image processing and fitting. <i>Note:</i> changing this may also change the values of <code>minimumGlintScanDensity</code> and <code>glintScanDensity</code> .	
<code>VPX_SendCommand("glintsegmentationMethod OvalFit");</code>	

14.7.6 Changes default setting for Freeze Feature	
GUI:	-none-
CLP Command:	freezeStops <i>Option</i> <i>Option: all, ImageDisplay</i>
SDK Function:	-none-
Default	All
Changes the default setting of stopping all video capture to just stopping the image display preview.	
<pre>VPX_SendCommand("freezeStops ImageDisplay"); VPX_sendCommand("freezeSops all");</pre>	

14.7.7 Toggle Freeze Video Imge Preview On / Off	
GUI:	Video > Freeze Video ^F EyeCamera window, SnowFlake button
CLP Command:	videoFreeze <i>BoolValue</i> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	<code>VPX_Video_Freeze (bool tf);</code>
Default:	No
Allows freezing the video image preview, Status window display and data collection. (c.f. freezeStops).	
<pre>VPX_SendCommand("videofreeze Yes");</pre>	

14.7.8 Reset Video Capture Device	
GUI:	Video > Reset Video
CLP Command:	videoReset
SDK Function:	<code>VPX_Video_Reset (void);</code>
Currently under considered for extension to accept a video stream identifier.	
Resets video capture device.	
<pre>VPX_SendCommand("videoReset");</pre>	

14.8 Calibration controls

14.8.1 Start Auto-Calibration	
GUI:	EyeSpace window, Auto-Calibrate / START Calibration button (toggle) ^A
CLP Command:	<code>calibrationStart</code>
SDK Function:	<code>VPX_AutoCalibrate();</code>
Starts the calibration process.	
<code>VPX_SendCommand ("calibrationStart");</code>	

14.8.2 Stop Auto-Calibration	
GUI:	EyeSpace window, Auto-Calibrate / STOP Calibration button (toggle) ^K
CLP Command:	<code>calibrationStop</code>
SDK Function:	<code>VPX_StopCalibration ();</code>
Stops the calibration process before its normal completion.	
<code>VPX_SendCommand ("calibrationStop");</code>	

14.8.3 Specify Calibration Stimulus Presentation Speed	
GUI:	EyeSpace window, Advanced button, Duration slider
CLP Command:	<code>calibration_StimulusDuration <i>milliseconds</i></code> <code><i>milliseconds</i> : integer between 1 and 400</code>
SDK Function:	<code>VPX_SetCalibrationSpeed (int <i>milliseconds</i>);</code> <code><i>milliseconds</i>: integer between 1 and 400</code>
Default:	<code>80 for Windows XP (depends upon the OS)</code>
<p>Specifies the delay in milliseconds between calibration stimulus changes (zoom rectangle decrements). <i>ViewPoint</i> attempts to set an optimal default value according to the operating system version. The "milliseconds" specification is only approximate and unfortunately varies between Microsoft operating system versions. This value also affects the warning time and the inter-stimulus interval (isi) that specify durations in units of stimulus duration milliseconds.</p> <p>The maximum duration is 200 ms, and the minimum is 1 ms.</p>	
<code>VPX_SendCommand ("Calibration_StimulusDuration 145");</code>	

14.8.4 Specify the duration of presentation of calibration warning notice	
GUI:	EyeSpace window, Advanced button, Warning slider
CLP Command:	calibration_WarningTime <i>durationUnits</i> <i>durationUnits: integer between 0 and 100</i>
SDK Function:	-none-
Default:	20
<p>Specifies the delay for posting a warning that calibration is about to start. <i>ViewPoint</i> attempts to set an optimal default value according to the operating system version. The time is only approximate. The delay is in units of the stimulus duration specified by the command: <code>calibration_StimulusDuration</code></p> <p>The value 0 specifies no warning is to be given.</p> <p>See also: Calibration_StimulusDuration</p>	
<pre>VPX_SendCommand ("Calibration_WarningTime 15");</pre>	

14.8.5 Specifies Interval Between Presentation of Calibration Stimulus Points	
GUI:	-none-
CLP Command:	calibration_ISI <i>intervalUnits</i> <i>intervalUnits: integer between 1 and 9</i>
SDK Function:	-none-
Default:	2
<p>Specifies the inter-stimulus interval between calibration points. <i>ViewPoint</i> attempts to set an optimal default value according to the operating system version. The time is only approximate. The delay is in units of the stimulus duration specified by the command: <code>calibration_StimulusDuration</code>.</p> <p>The value 0 specifies no ISI time.</p> <p>See also: calibration_StimulusDuration</p>	
<pre>VPX_SendCommand ("Calibration_ISI 2");</pre>	

14.8.6 Calibration Snap Mode	
GUI:	EyeSpace window, Advanced button, Snap Mode checkbox
CLP Command:	<code>calibration_SnapMode BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	<code>VPX_set_snapCalibrationMode(bool tf);</code>
Default:	No
<p>Snap mode means that the calibration stimulus images (the zooming concentric rectangles) are not presented, and the calibration of the currently selected point is immediately performed based on the current eye position. This is useful for remote or manual calibration as when using the scene camera option and when controlling calibration with stimulus points are controlled and generated on a remote computer; and also for use with a scene camera.</p> <p>This command affects the behavior of the "Re-Present" and the "Slip-Correction" buttons/commands and is indicated by the appearance of an asterisk (*) on these buttons when in this special mode.</p> <p>It is not necessary to enter this mode if using the command: <code>calibration_Snap</code>.</p> <p>See also:</p> <ul style="list-style-type: none"> <code>calibration_Snap</code> <code>calibration_AutoIncrement</code> <code>calibrationRedoPoint</code> 	
<code>VPX_SendCommand ("calibration_SnapMode ON");</code>	

14.8.7 RePresent in Snap Calibration Mode	
GUI:	EyeSpace window, Re-Present * (when asterisk is showing)
CLP Command:	<code>calibration_Snap</code>
SDK Function:	<code>-none-</code>
Default:	
<p>Snap mode means that the calibration stimulus images (the zooming concentric rectangles) are not presented, and the calibration of the currently selected point is immediately performed based on the current eye position.</p> <p>This is useful for remote or manual calibration as when using the scene camera option and when controlling calibration with stimulus points are controlled and generated on a remote computer; and also for use with a scene camera.</p> <p>Use of this command does not require, and in may situations obviates the need for, changing the <code>calibration_SnapMode</code>.</p>	
<code>VPX_SendCommand ("calibration_Snap");</code>	

14.8.8 AutoIncrement Calibration Mode	
GUI:	EyeSpace window, Advanced button, Auot-Increment checkbox
CLP Command:	Calibration_AutoIncrement BoolValue <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
It affects the behavior of the "Re-Present" button/command and is indicated by a double plus (++) on this button when in this special mode. This mode is useful for manual calibration as with a scene camera.	
VPX_SendCommand ("calibration_AutoIncrement TRUE ");	

14.8.9 Calibration Stimulus Point Presentation Order	
GUI:	EyeSpace window, Advanced button, Presentation Order pulldown menu
CLP Command:	calibration_PresentationOrder orderChoice <i>orderChoice: Sequential, Random, Custom</i>
SDK Function:	-none-
This affects both auto-callibration and manual calibration that sequence through the index values. If the user has set: calibration_PresentationOrder Sequential then the index value and the calibration stimulus point number are the same.	
VPX_SendCommand ("calibration_PresentationOrder Random");	

14.8.10 Specify Number of Calibration Stimulus Points	
GUI:	EyeSpace window, drop-down menu: 6 ... 72
CLP Command:	calibration_Points numberOfPoints <i>numberOfPoints : integer from the following set: { 6, 9, 12, 16, 20, 25, 30, 36, 42, 49, 56, 64, 72 }</i>
SDK Function:	VPX_SetCalibrationPoints (int numberOfPoints); <i>numberOfPoints : integer from the following set: { 6, 9, 12, 16, 20, 25, 30, 36, 42, 49, 56, 64, 72 }</i>
Default:	16
Sets the number of calibration points to be presented. Note that the valid values are either N x N or N x (N-1)	
VPX_SendCommand ("calibration_Points 12");	

14.8.11 Specify Calibration Stimulus Point Color	
GUI:	EyeSpace window, Advanced button, Set Stimulus Color button
CLP Command:	<code>calibration_stimulusColor ColorValue</code> <i>ColorValue: Red 0-255, Green 0-255, Blue 0-255</i>
SDK Function:	-none-
Specifies the red, green and blue components (0 to 255) of the calibration stimulus points E.g. , 255 255 255 is white and 100 100 255 is a sky blue.	
VPX_SendCommand ("calibration_StimulusColor 255 255 025");	

14.8.12 Specify Calibration Stimulus Window Background Color	
GUI:	EyeSpace window, Advanced button, Set Background Color button
CLP Command:	<code>calibration_BackgroundColor ColorValue</code> <i>ColorValue: Red 0-255, Green 0-255, Blue 0-255</i>
SDK Function:	-none-
Specifies the red, green and blue components (0 to 255) of the calibration Stimulus window background E.g. , 255 255 255 is white and 100 100 255 is a sky blue.	
VPX_SendCommand ("calibration_BackgroundColor 050 100 255");	

14.8.13 Randomize Calibration Stimulus Points Check Box (DEPRECATED)	
GUI:	EyeSpace window, Advanced button, Randomize Calibration checkbox
CLP Command:	<code>calibration_randomize BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
Default:	Yes
DEPRECATED calibration_Randomize ON → calibration_PresentationOrder Random calibration_Randomize OFF → calibration_PresentationOrder Sequential <i>See also:</i> <code>calibration_PresentationOrder</code>	
VPX_SendCommand ("calibration_randomize ON");	

14.8.14 Specify Calibration Stimulus Point Presentation Order	
GUI:	-none-
CLP Command:	<code>calibration_CustomOrderList n1 n2 n3 n4 n5 n6 ...</code>
SDK Function:	-none-
Default:	<code>6 5 4 3 2 1 9 8 7 12 11 10 16 15 14 13 20 19 18 17 21 22 23 ...</code>
<p>Allow specification of up to 72 calibration point in the desired order of presentation.</p> <p><i>Caution:</i> An error will occur if the user has provided a point number in the list, which is larger than the selected number of calibration Points (i.e., the quantity of calibration points). For example, if the user specified: customOrderList 6 72 4 69 2 1 and also specified: calibrationPoints 6, the points 72 and 69 would cause errors, because they are greater than 6.</p> <p>See also:</p> <ul style="list-style-type: none"> <code>calibration_SelectIndex</code> <code>calibration_SelectPoint</code> <code>calibrationPoints</code> 	
<pre>VPX_SendCommand ("calibration_ CustomOrderList 6 1 5 2 4 3");</pre>	

14.8.15 Specify Individual Custom Calibration Stimulus Points	
GUI:	-none-
CLP Command:	<code>calibration_CustomOrderEntry index calStimPoint</code> <code>index : integer in { 1 to N }</code> <code>calStimPoint : integer in { 1 to N }</code>
SDK Function:	-none-
Default:	See <code>calibration_CustomOrderList</code> , just above.
<p>Used to specify individual custom calibration point entries.</p> <p><i>Caution:</i> see the Caution section in calibration_CustomOrderList, above.</p> <p>See also:</p> <ul style="list-style-type: none"> <code>calibration_SelectIndex</code> <code>calibration_SelectPoint</code> 	
<pre>VPX_SendCommand ("calibration_CustomOrderEntry 1 6");</pre>	

14.8.16 Display Custom Calibration Stimulus Point Order	
GUI:	-none-
CLP Command:	calibration_CustomOrderDump
SDK Function:	-none-
Default:	
Prints the customOrder of calibration points to the History window.	
<code>VPX_SendCommand ("calibration_CustomOrderDump");</code>	

14.8.17 Select the Specified Calibration Data Point	
GUI:	EyeSpace window, Data Point slider or/ Click mouse in EyeSpace window, calibration graphics well
CLP Command:	calibration_SelectPoint <i>pointSelection</i> <i>pointSelection: LAST, NEXT, or int { 1 .. N }</i>
SDK Function:	VPX_EyeSpace_SelectPoint (<i>int SelectPoint</i>);
Selects the specified calibration data point. The point will be highlighted in the EyeSpace window. <i>See also:</i> calibration_SelectIndex	
<code>VPX_SendCommand ("calibrationSelectPoint 7");</code>	

14.8.18 Select the Index Number that Maps to the Specified Calibration Data Point	
GUI:	-none
CLP Command:	<code>calibration_SelectIndex <i>indexSelection</i></code> <i>indexSelection</i> : <i>LAST, NEXT, or int { 1 .. N }</i>
SDK Function:	-none-
<p>Selects the index number that maps to a calibration stimulus point. The corresponding stimulus point will be highlighted in the EyeSpace window and the calibration data point slider will be adjusted.</p> <p>When the calibration_PresentationOrder is Sequential the selected index and the selected point are the same. When calibration_PresentationOrder is Random or Custom, the index is incremented and the selected point is taken from the random, or custom, table.</p> <p><i>Note:</i> if the user has set: <code>calibration_PresentationOrder Random</code> then the series is re-randomized every time the set finishes, so that there is a new set for the next loop.</p> <p><i>See also:</i></p> <ul style="list-style-type: none"> <code>calibration_SelectPoint</code> <code>calibration_PresentationOrder</code> <code>calibration_CustomList</code> <code>calibration_CustomOrderEntry <i>index calStimPoint</i></code> 	
<pre>VPX_SendCommand ("calibration_SelectIndex 7"); // look up Pt# in CustomList</pre>	

14.8.19 Undo the last operation on a Calibration Data Point	
GUI:	EyeSpace window, Undo button
CLP Command:	<code>calibrationUndo</code>
SDK Function:	<code>VPX_calibUndo</code>
<p>Re-centers the selected calibration point.</p>	
<pre>VPX_SendCommand ("calibrationUndo");</pre>	

14.8.20 Re-Present the Specified Calibration Data Point	
GUI:	EyeSpace window, Re-present button
CLP Command:	<code>calibrationRedoPoint CalibrationPoint</code> <i>CalibrationPoint: Numbered point to select.</i>
SDK Function:	<code>VPX_ReCalibratePoint (int SelectPoint);</code>
<p>Re-presents the specified calibration data point.</p> <p>The behavior is determined by the <code>calibration_SnapMode</code> and <code>calibration_AutoIncrement</code> specifications.</p> <p>The <i>CalibrationPoint</i> parameter is optional, if omitted, the currently selected calibration point will be re-presented.</p> <p><i>Note:</i> the point number corresponds to the slider values in the EyeSpace window and does not depend on the calibration method chosen.</p> <p><i>See also:</i></p> <ul style="list-style-type: none"> <code>calibration_SnapMode</code> <code>calibration_AutoIncrement</code> 	
<pre>VPX_SendCommand ("calibrationRedoPoint 7"); VPX_SendCommand ("calibrationRedoPoint"); // defaults to current point</pre>	

14.8.21 Specify Custom Calibration Stimulus Point Locations	
GUI:	-none-
CLP Command:	<code>calibration_CustomPoint 2 0.95 0.95</code>
SDK Function:	<code>VPX_SendCommand("calibration_CustomPoint %d %g %g", pointIndex, loc.x, loc.y);</code>
<p>Specifies custom locations of the calibration stimulus points. Note: the nearest-neighbor grid-lines in the EyeSpace are not automatically draw when this option is used, because the points could be in any configuration.</p> <p><i>See also:</i></p> <ul style="list-style-type: none"> <code>calibration_CustomPointsUsed</code> <code>calibration_CustomPointDump</code> 	
<pre>VPX_SendCommand ("calibration_CustomPoint 2 0.95 0.95");</pre>	

14.8.22 Turn Custom Calibration Stimulus Point Location ON / OFF	
GUI:	-none-
CLP Command:	<code>calibration_CustomPointsUsed BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	<code>VPX_SendCommand("calibration_CustomPoint %d %g %g", pointIndex, loc.x, loc.y);</code>
Turns custom calibration stimulus point location on or off. <i>See also:</i> <code>calibration_CustomPoint</code> <code>calibration_CustomPointDump</code>	
<code>VPX_SendCommand ("calibration_CustomPointsUsed YES");</code>	

14.8.23 Print Locations of custom calibration stimulus points in EventHistory window	
GUI:	-none-
CLP Command:	<code>calibration_CustomPointDump</code>
SDK Function:	<code>VPX_SendCommand("calibration_CustomPointDump");</code>
Prints the custom calibration point list in the History window.. <i>See also:</i> <code>calibration_CustomPoint</code> <code>calibration_CustomPointsUsed</code>	
<code>VPX_SendCommand ("calibration_CustomPointDump");</code>	

14.8.24 Controls display of nearest-neighbor gridlines in the EyeSpace window	
GUI:	-none-
CLP Command:	<code>calibration_showEyeSpaceGrid BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	<code>VPX_SendCommand("calibration_showEyeSpaceGrid");</code>
Turns ON or OFF the "spider web" nearest-neighbor grid between calibration points in the EyeSpace window. <i>See also:</i> <code>calibration_CustomPointsUsed</code>	
<code>VPX_SendCommand ("calibration_showEyeSpaceGrid");</code>	

14.8.25 Compensate for Slip	
GUI:	EyeSpace window, Slip-Correction button
CLP Command:	calibrationSlipCorrection CalibrationPoint <i>CalibrationPoint: Numbered point to select.</i>
SDK Function:	VPX_SlipCorrection (int SelectPoint);
<p>Re-presents the specified calibration data point and re-aligns calibration to compensate for slip. Normally a point near the center of the display should be chosen.</p> <p>See also: calibration_SnapMode</p>	
<pre>VPX_SendCommand ("calibrationSlipCorrection 7");</pre>	

14.8.26 Adjust Calibration Area	
GUI:	Controls window, Regions Tab, Calibration Region radio button
CLP Command:	calibrationRealRect L T R B <i>L T R B: Normalized floating point values for the corners.</i>
SDK Function:	-none-
Default:	0.1 0.1 0.9 0.9
<p>Allows the user to adjust the calibration area (size and position) within which the calibration stimulus points are presented. The four values are the floating point coordinates (0.0 – 1.0) of the bounding rectangle, listed in the order: Left, Top, Right, Bottom.</p>	
<pre>VPX_SendCommand ("calibrationRealRect 0.2 0.2 0.8 0.8");</pre>	

14.8.27 Save Image of Eye at each Calibration Data Point	
GUI:	-none-
CLP Command:	calibration_SaveEyeImages BoolValue <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
Default:	off
<p>When enabled this command will take a snapshot of the eye during calibration stimulus presentation at each calibration point. This feature is useful to examine the position of the eye when presented the stimulus. This allows users to determine any problems with subjects eyes or loss of a direct image of the pupil at a certain calibration point.</p>	
<pre>VPX_SendCommand ("calibration_SaveEyeImages yes");</pre>	

14.9 Controls: Criteria Controls

14.9.1 Specify amount of Smoothing	
GUI:	Controls window, DataCriteria tab, Smoothing Points slider
CLP Command:	<code>smoothingPoints <i>IntValue</i></code>
SDK Function:	<code>VPX_SetSmoothing (<i>IntValue</i>);</code>
Default:	4
<p>The amount of smoothing to apply to the gaze position. The number specifies the number of previous sample point to use in the trailing average. A value of 4 makes attractive and useful real-time graphics. The value 1 indicates to use only the current point, i.e., no smoothing.</p> <p>See also: <code>smoothingMethod</code>, <code>velocityCriterion</code></p> <p>Modified: 2.8.2.44</p>	
<pre>VPX_SendCommand ("smoothingPoints 3");</pre>	

14.9.2 Specify Smoothing Algorithm to Apply	
Control or Menu:	-none-
CLP Command:	<code>smoothingMethod <i>method</i></code> <code>method: SMA, EMA</code>
SDK Function:	-none-
Default:	SMA
<p>The user may choose between two smoothing algorithms: Simple Moving Average (SMA) and Exponential Moving Average (EMA).</p> <p>The SMA method uniformly averages N pointsBack, i.e., all points having equal weight.</p> $SMA(t) = [x(t) + x(t-1) + \dots + x(t-n)] / N ; \text{ where } n = (N-1)$ <p>The EMA method uses the following algorithm:</p> $EMA(t) = (\text{currentValue} - EMA(t-1)) * K + EMA(t-1) ; \text{ where } K = 2 / (\text{pointsBack} + 1) .$ <p>The number of pointsBack is adjusted by the Smoothing slider or the CLP command <code>smoothingPoints N</code>.</p> <p>Note: The successfully set method is currently reported in the Event History window.</p> <p>See also: <code>smoothingPoints</code></p> <p>Added: 2.8.2.44</p>	
<pre>VPX_SendCommand ("smoothingMethod EMA");</pre>	

14.9.3 Specify Velocity Threshold	
Control or Menu:	Controls window, DataCriteria tab, Saccade Velocity slider
CLP Command:	velocityCriterion <i>NormalizedValue</i> velocityThreshold <i>NormalizedValue</i> <i>NormalizedValue: floating point number in range 0.0 to 1.0</i>
SDK Function:	-none-
Default:	0.10
<p>The velocity level that is used to distinguish between saccades and fixations. This criterion threshold value is displayed in the penplot window plot for total velocity.</p> <p>Note that the velocity magnitude, and consequently the required threshold, will be affected by the amount of smoothing applied.</p> <p><i>See also:</i></p> <p>smoothingPoints driftCriterion</p>	
<pre>VPX_SendCommand ("velocityCriterion 0.8"); // Preferred VPX_SendCommand ("velocityThreshold 0.45"); // OlderForm</pre>	

14.9.4 Specify amount of Drift Allowed	
Control or Menu:	Controls window, DataCriteria tab, Fixation Drift Allowed slider
CLP Command:	driftCriterion <i>NormalizedValue</i> <i>NormalizedValue: floating point number in range 0.0 to 1.0</i>
SDK Function:	-none-
Default:	0.03
<p>Specifies the absolute drift away from the fixation start point, or the last drift start point, that is tolerated before a Drift classification is made. The units are in normalized stimulus window coordinates, just like gaze point.</p> <p>Without a Drift Criterion, the eyes can slowly change position and still be classified as a Fixation, because the velocity never exceeded the Saccade Velocity Criterion.</p> <p>This criterion threshold value is displayed in the penplot window plot for for Drift.</p> <p><i>See also:</i></p> <p>velocityCriterion penPlot +DRIFT</p> <p><i>Changes:</i></p> <p>Default value changed in version 2.8.3 from 0.1 to 0.03</p>	
<pre>VPX_SendCommand ("driftCriterion 0.025");</pre>	

14.9.5 Specify Pupil Aspect Ratio Failure Criterion	
GUI:	Controls window, DataCriteria tab, Pupil Aspect Criterion
CLP Command:	<code>pupilAspectCriterion <i>NormalizedValue</i></code> <i>NormalizedValue</i> : a floating point number in range 0.0 to 1.0
SDK Function:	<code>VPX_SetPupilOvalCriteria (<i>NormalizedValue</i>);</code>
Default:	0.05
<p>Specifies the aspect ratio at which the data quality marker indicates that there is a problem. The default value is 0.05 , so that any aspect ratio is accepted. Typically a useful value is about 0.8 Adjust the pupil oval ratio criteria for classification as the pupil. Range 0.0 – 1.0. The pupil oval fit changes color from yellow to orange when criterion is violated.</p>	
<pre>VPX_SendCommand ("pupilAspectCriterion 0.8");</pre>	

14.9.6 Specify Pupil Width Failure Criterion	
GUI:	Controls window, DataCriteria tab, Maximum Pupil Width slider
CLP Command:	<code>pupilMaxWidthCriterion <i>NormalizedValue</i></code> <i>NormalizedValue</i> : a floating point number in range 0.0 to 1.0
SDK Function:	-none-
Default:	0.75
<p>Specifies the pupil width at which the data quality marker indicates that there is a problem.</p>	
<pre>VPX_SendCommand ("pupilMaxWidthCriterion 0.35");</pre>	

14.10 Region of Interest (ROI)

14.10.1 Define an ROI Box	
GUI:	-none-
CLP Command:	<code>setROI_RealRect <i>Index L T R B</i></code> <i>Index: Integer value indicating the ROI to specify.</i> <i>L T R B: Normalized floating point values for the corners.</i>
SDK Function:	<code>int VPX_SetROI_RealRect (int n, RealRect rr)</code>
Default:	One box the center and eight iso-eccentric boxes.
<p>Defines a Region of Interest (ROI) (aka window discriminator box) The first value <n> specifies which ROI to adjust. The next four values are the normalized (0.0 to 1.0) coordinates of the bounding rectangle.</p> <p>See also:</p> <pre> setROI_Alloff setROI_isoEccentric VPX_GetROI_RealRect(n,rr); VPX_drawROI(HWND hWnd, int activeRegion); </pre>	
<pre> VPX_SendCommand ("setROI_RealRect 5 0.1 0.1 0.9 0.9"); // Sets ROI #5 with 10% margins </pre>	

14.10.2 Specify Number of ROI to be drawn in a circle around center of window	
GUI:	-none-
CLP Command:	<code>setROI_isoEccentric <i>NumberOfBoxes</i></code> <i>NumberOfBoxes: integer number of ROI in a circle.</i>
SDK Function:	<code>VPX_SetROI_isoEccentric (int numberOfBoxes);</code>
<p>Specifies the number of ROI boxes to be drawn in an isoeccentric distribution, i.e., in a circle around the center of the window. This also clears previous ROI box settings and refreshes the displays.</p>	
<pre> VPX_SendCommand("setROI_isoEccentric 15"); </pre>	

14.10.3 Remove all ROI Boxes	
GUI:	Controls window, DataDisplay tab, ROI Regions checkbox
CLP Command:	<code>setROI_AllOff</code>
SDK Function:	-none-
Removes all ROI boxes.	
<code>VPX_SendCommand("setROI_AllOff");</code>	

14.10.4 Select a Specific ROI	
GUI:	Controls window, Regions tab, Region: slider
CLP Command:	<code>setROI_Selection ROIbox</code> ROIbox : integer number of the box to be selected. i.e. 1 or 2 or 3 ect. Up to the max number of ROI boxes enabled.
SDK Function:	-none-
Default:	-none-
Select ROI number N . The selected ROI is shown in red when the ROI overlay graphics is shown. This can be used to highlight one particular ROI. See also: "setROI_Lock"	
<code>VPX_SendCommand("setROI_Selection 9");</code>	

14.10.5 Select the next ROI Box	
GUI:	Controls window, Regions tab, Region: slider F8
CLP Command:	<code>setROI_selectNext</code>
SDK Function:	-none-
Select the next ROI number. This can be used to highlight an area.. (current + 1). The selected ROI is drawn in red when the ROI overlay graphics is shown. This can be used to highlight an area. HINT: A mouse wheel is extremely helpful for moving between selections of the region slider.	
<code>VPX_SendCommand("setROI_selectNext");</code>	

14.10.6 Lock ROI Settings	
GUI:	Controls window, Regions tab, Region: slider, None
CLP Command:	<code>setROI_Lock</code>
SDK Function:	<code>-none-</code>
Deselects all ROI, i.e. no ROI is selected. See also: " <code>setROI_Selection N</code> "	
<code>VPX_SendCommand("setROI_Lock");</code>	

14.11 PenPlot controls

14.11.1 Specify Which PenPlot Traces to Display	
GUI:	Menu : PenPlots >
CLP Command:	<code>penPlot +chartItemName - chartItemName</code> <code>chartItemName : any of the following:</code> <code>All, Tming, Vergence, Xvelocity, Yvelocity, Tvelocity, Width,</code> <code>Aspect, Xgaze, Ygaze, Xangle, Yangle, Torsion, Seconds,</code> <code>HeadPosition, HeadAngle, Xpupil, Ypupil, Xglint, Yglint,</code> <code>Drift, Events, fixationTime, Timing, ROI, Quality</code>
SDK Function:	<code>-none-</code>
Default:	<code>+Xgaze +Ygaze +Velocity +Aspect +Drift</code>
<p>Specifies which penPlots to display in the PenPlot window. To include a plot precede its name string with a "+", to exclude it precede its name with "-". Be careful that there is no white space between + or - and the string. Separate multiple strings arguments on the same line with a space, as in the example below.</p> <pre>+Velocity +Width +Aspect +Timing +Vergence (data available with binocular option only) +HeadPosition (data available with head track option only) +HeadAngle (data available with head track option only) +xPupilPoint +yPupilPoint (raw eyespace pupil position data as a function of time) +XAngle +YAngle (data valid only when accurate GeometryGrid measurements are set)</pre>	
<code>VPX_SendCommand("penPlot +Xgaze +Ygaze -Tvelocity +Width +Aspect -Timing");</code>	

14.11.2 BackGround Color of PenPlot Traces	
GUI:	-none-
CLP Command:	<code>penPlot_BackgroundColor ColorValue</code> <i>ColorValue: Red 0-255, Green 0-255, Blue 0-255</i>
SDK Function:	-none-
Default:	160, 160, 160
Controls of background color for penPlot stripChart windows. See also: <code>penPlot_LimenFillColor</code>	
<code>VPX_SendCommand("penplot_BackgroundColor 115 100 235");</code>	

14.11.3 PenPlot Back Ground Color	
GUI:	-none-
CLP Command:	<code>penPlot_LimenFillColor ColorValue</code> <i>ColorValue: Red 0-255, Green 0-255, Blue 0-255</i>
SDK Function:	-none-
Default:	144, 144, 144
Specifies the color that fills the rectangle within the criterion lines (threshold, limen) in the penPlots. The default value is slightly darker gray than the default penPlot_BackgroundColor. See also: <code>penPlot_BackgroundColor</code>	
<code>VPX_SendCommand("penplot_LimenFillColor 115 100 235");</code>	

14.11.4 Specify Speed of PenPlot Scrolling	
GUI:	Menu : PenPlots > Speed > { ¼ x, ½ x, 1x, 2x, 4x, 8x, 16x }
CLP Command:	<code>penPlot_Speed xInc</code> <i>xInc: float</i>
SDK Function:	-none-
Default:	2
Specifies the number of pixels to increment the the penPlot along the x (time) axis. Sometimes it is desirable to increase the speed of the penPlot scrolling, so that details, markers, and classifications can be more easily distinguished. Modified: 2.8.2.51	
<code>VPX_SendCommand("penPlot_speed 6");</code>	

14.11.5 Specify Range of PenPlot Values	
GUI:	-none-
CLP Command:	<pre>penPlot_Range plotName lowerValue upperValue plotName: any of: XGaze, YGaze, Drift, Fix, Vergence, Xvelocity, Yvelocity, Tvelocity, Width, Aspect, XAngle, YAngle, Torsion, HeadPosition, HeadAngle, XPUPIL, YPUPIL, XGLINT, YGLINT, Vergence lowerValue, upperValue: float</pre>
SDK Function:	-none-
Default:	varies depending on plotName
Specifies the lower and upper plot range values for a particular penPlot graph.	
Added: 2.8.2.51	
<pre>VPX_SendCommand("penPlot_Range xgaze 0.2 0.6");</pre>	

14.11.6 Specify the behavior of the penpot after a video freeze	
GUI:	-none-
CLP Command:	<pre>penPlot_restartAfterFreeze BoolValue BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</pre>
SDK Function:	-none-
Default:	No
Specifies the behavior of the penpot after a video freeze. Specifies whether or not the pen will continue going from where it is, or if the penplot window will be erased and the pen will reset and start afresh from the left.	
<pre>VPX_SendCommand("penplot_restartafterfreeze Yes");</pre>	

14.12 Graphics controls

14.12.1 Specify the color of the GazeSpace and PenPlot Lines	
GUI:	Controls window, DataDisplay tab, Eye A button or , Eye B button
CLP Command:	<pre>penColorA <i>ColorValue</i> penColorB <i>ColorValue</i> <i>ColorValue</i>: Red 0-255, Green 0-255, Blue 0-255</pre>
SDK Function:	-none-
Default:	penColorA: 100 255 0 (limeGreen); penColorB : 0 255 255 (cyan)
<p>Specifies the red, green and blue components (0 to 255) of the gazespace and penPlot lines. E.g. , 255 255 255 is white and 100 100 255 is a sky blue.</p>	
<pre>VPX_SendCommand("penColorA 115 100 235"); VPX_SendCommand("penColorB 0 255 0"); // Green</pre>	

14.12.2 Specify which Overlay Graphics to Display in the GazeSpace Window	
GUI:	Controls window, DataDisplay tab, Gaze option checkbox matrix
CLP Command:	<pre>gazeGraphicsOptions <i>+graphicsOptions -graphicsOptions</i> <i>graphicsOptions</i> : +ROI +POG +Path +Fix +Size +Grid +Cal +Raw</pre>
SDK Function:	-none-
Default:	+ROI +Path +Image
<p>Specifies which overlay graphics to display in the GazeSpace window. Use -graphicsOption to exclude an option, or use +graphicsOption to include an option. Note: The +/- must be next to the key work, i.e., NOT separated from it by a space. Separate multiple arguments by spaces, as in the example here below. See also: stimulusGraphicsOptions</p>	
<pre>VPX_SendCommand("gazeGraphicsOptions +POG -Raw -ROI");</pre>	

14.12.3 Specify which Overlay Graphics to Display in the Stimulus Window	
GUI:	Controls window, DataDisplay tab, checkbox matrix
CLP Command:	<code>stimulusGraphicsOptions +graphicsOptions -graphicsOptions graphicsOptions : +ROI +POG +Path +Fix +Size +Grid +Cal +Raw +Image</code>
SDK Function:	<code>-none-</code>
Default:	<code>+POG +Image</code>
<p>Specifies which overlay graphics to display in the Stimulus window. Use -graphicsOption to exclude an option, or use +graphicsOption to include an option. Note: The +/- must be next to the key work, i.e., NOT separated from it by a space. Separate multiple arguments by spaces, as in the example here below. See also: gazeGraphicsOptions</p>	
<pre>VPX_SendCommand("stimulusGraphicsOptions +POG -Raw -ROI");</pre>	

14.12.4 Erase Data Displays in the GazeSpace and Stimulus windows	
GUI:	Controls window, DataDisplay tab, Erase Display button
CLP Command:	<code>eraseDisplays</code>
SDK Function:	<code>-none-</code>
<p>Clears the previous drawn data from the GazeSpace and Stimulus windows. See also: <code>timedErase_autoErase</code> <code>timedErase_delaySeconds</code></p>	
<pre>VPX_SendCommand("eraseDisplays");</pre>	

14.12.5 Automatically erase display windows	
GUI:	-none-
CLP Command:	timedErase_autoErase BoolValue BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle
SDK Function:	-none-
Default:	off
Automatically erase display windows. <i>See also:</i> timedErase_delaySeconds eraseDisplays	
<pre>VPX_SendCommand("timedErase_autoErase On"); VPX_SendCommand("timedErase_delaySeconds 6.5");</pre>	

14.12.6 Specify time delay for auto erase	
GUI:	-none-
CLP Command:	timedErase_delaySeconds seconds seconds : floating point number
SDK Function:	-none-
Default:	off
Automatically erase display windows. <i>See also:</i> timedErase_autoErase eraseDisplays	
<pre>VPX_SendCommand("timedErase_autoErase On"); VPX_SendCommand("timedErase_delaySeconds 6.5");</pre>	

14.13 Stimulus Window controls

14.13.1 Specify Stimulus Source	
GUI:	<code>Stimuli > View Source > { StimulusWindow, SceneCamera, Interactive Computer Display ... }</code>
CLP Command:	<code>viewSource options</code> <code>options: StimulusWindow, SceneCamera, 1, 2, 3</code>
SDK Function:	-none-
Default:	Yes
Specify the type of stimulus the subject will be viewing.	
<code>VPX_SendCommand("viewSource SceneCamera");</code>	

14.13.2 Specify Custom Stimulus window Size and Position

GUI: **1st) select style Stimuli> Stimulus Window Properties > Normal Adjustable Window**
 2nd) resize the window by dragging its border
 3rd) change style: Stimuli> Stimulus Window Properties > Custom Static Position

CLP Command: **stimWind_CustomStatic L T R B**
 L T R B: Normalized floating point values for the physical dimensions.

SDK Function: ~~VPX_StimWind_CustomStatic ("int x1, int y1, int x2, int y2");~~

Specifies a custom position and size of the **Stimulus** window. Where L,T,R,B correspond to the left, top, right and bottom physical dimensions in the display space. The following example assumes that we have a secondary display placed in the virtual desktop with the tops of the two displays at the same level, namely zero. Further, the primary monitor is 1280 x 1024 and the secondary is 1024 x 768, such that the top left origin of the primary monitor is at (0,0), the top left origin of the secondary monitor is at (1280,0), and the extreme diagonal bottom right point is at (1280+1024,768) or (2304,768) **stimWind_CustomStatic 1280 0 2304 768**.

Sets the custom static **Stimulus** window in the position specified. Displays the **Stimulus** window as a static window (no borders or title bar). The window will be always in front. This is useful if the graphics card does not show a second display.

This neither (i) switches the display selection to Custom Static, nor (ii) causes the stimulus window to be shown.

Note: obsolete term 'customStimulusWindow' would raise the window.

See also:

stimWind_FullDisplay Custom
setWindow Stimulus Show
stimulus_ImageHidden

```
VPX_SendCommand( "stimWind_CustomStatic 1280 0 2304 768" );
VPX_SendCommand( "setWindow Stimulus Show" );
```

14.13.3 Automatically Show the Stimulus Window on Primary Monitor	
GUI:	Stimuli> Stimulus Window Properties > AutoShow on Calibrate
CLP Command:	<code>stimwind_AutoShowOnCalibrate BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	<code>VPX_StimWind_AutoShowOnCalibrate (bool tf);</code>
Default:	Yes
<p>ViewPoint has been designed to make calibration easy and fast, even when there is only one computer display, and when the experimenter wants to do self-testing.</p> <p>If (i) the <i>AutoShowOnCalibrate</i> feature is active, (ii) the stimulus window is set to display full screen on the primary monitor, and (iii) the stimulus window is hidden, then ViewPoint will automatically show (unhide) the stimulus window, whenever a calibration start procedure starts.</p> <p>See also:</p> <p><code>stimWind_FullDisplay 1</code></p>	
<code>VPX_SendCommand("stimwind_autoshownoncalibrate No");</code>	

14.13.4 Specify How and where to show Stimulus window	
GUI:	Stimuli> Stimulus Window Properties > Normal Adjustable Stimuli> Stimulus Window Properties > Custom Static Stimuli> Stimulus Window Properties > Full Screen Monitor 1 (Primary) Stimuli> Stimulus Window Properties > Full Screen Monitor 2 Stimuli> Stimulus Window Properties > Full Screen Monitor 3 etc.
CLP Command:	<code>stimWind_FullDisplay <i>OptionString</i></code> <i>MonitorNumber</i> : integer <i>OptionString</i> : Adjustable, Custom, 1, Primary, 2, Secondary
SDK Function:	<code>VPX_StimWind_FullDisplay (int monitor)</code> <code>VPX_StimWind_Adjustable(void)</code>
Default:	1
<p>Specifies where and how to display the stimulus window. Arguments Primary, 1, Secondary, 2, etc. specify that the stimulus window will be displayed full screen on these monitors. Adjustable indicates that the stimulus window is to be displayed as a regular adjustable-size floating window. Custom was primarily developed to help users who have dual monitor display card that do not correctly tell the Windows OS that there are two devices; these are now rare.</p> <p><i>Note 1 :</i> This command ONLY specifies the selection of the device on which the stimulus window will be displayed (or is displayed, if the stimulus window is currently showing), it does NOT cause the stimulus window to be shown. There are advantages to separating these instructions for (i) selecting the device and for (ii) showing / hiding the window, not the least of which is isomorphism to the conceptual layout of the GUI menu item groupings.</p> <p><i>Note 2:</i> In previous versions, prior to 2.8.3, the argument 0 corresponded to the instruction to Hide the stimulus window, or in some previous versions, to toggle Show / Hide. Argument 0 should no longer be used and its effect may be unpredictable.</p> <p><i>See also:</i> <code>setWindow STIMULUS SHOW</code> <code>setWindow STIMULUS HIDE</code> <code>stimWind_CustomStatic</code> <code>stimWind_AutoShowOnCalibrate</code></p> <p><i>Modified: 2.8.2.52</i></p>	
<code>VPX_SendCommand("stimWind_FullDisplay Secondary");</code>	

14.13.5 Calibrate to a third party application window	
GUI:	-none-
CLP Command:	-none-
SDK Function:	<code>VPX_SetExternalStimulusWindow(HWND hWnd);</code>
<p>This provides a mechanism for <i>ViewPoint</i> to draw the calibration stimuli directly into a window created by another application. The argument to this function is the handle of the created window.</p> <p>CAUTION: When finished, the programmer must make certain to call this function again with a NULL HWND argument (or the handle of yet another window), before destroying the created window.</p> <p>The calibration stimuli are still drawn in the <i>ViewPoint</i> Stimulus and GazeSpace windows, as usual.</p> <p>See same code in: <code>VPX_MFC_Demo.cpp</code></p> <p>Contrast to: <code>HWND VPX_GetViewPointStimulusWindow();</code></p> <pre>CWnd* pWnd = GetDlgItem(IDC_StimulusPicture) ; HWND hWnd = pWnd->GetSafeHwnd() ; VPX_SetExternalStimulusWindow(hWnd);</pre>	

14.14 Window related controls

14.14.1 Print ViewPoint windows	
GUI:	File > Print > * window
CLP Command:	<code>printWindow <i>windowNameString</i></code> <i>windowNameString</i> : Main, EyeCamera, EyeSpace, Controls, Status, GazeSpace, PenPlot
SDK Function:	-none-
<p>Using this command, windows can be sent to a printer. Note: you may want to select Freeze before Print to prevent pull down menu occlusion.</p> <p>See also: <code>printDateTimeStamp</code></p> <pre>VPX_SendCommand("printWindow EyeCamera");</pre>	

14.14.2 Include Date and Time Stamp on Printed windows	
GUI:	File > Print > Date Timestamp printouts (toggle)
CLP Command:	<code>printDateTimeStamp <i>BoolValue</i></code> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK Function:	-none-
Default:	On
<p>It is often useful to have the current date and time stamp included on printouts. Use this command to turn this feature on or off.</p> <p>See: printWindow</p>	
<pre>VPX_SendCommand("printDateTimeStamp Off");</pre>	

14.14.3 Move and Resize Window	
GUI:	Drag window with mouse, resize window with mouse.
CLP Command:	<code>moveWindow <i>windowNameString</i> <i>L T R B</i></code> <i>windowNameString</i> : Main, EyeCamera, EyeSpace, Controls, Status, GazeSpace, PenPlot <i>L T R B</i> : four integers describing the left, top, right and bottom corners of the window rectangle.
SDK Function:	-none-
<p>Both moves and resizes a particular window to the defined coordinates.</p> <p>See also: setWindow</p>	
<pre>RECT r = { 0, 0, 500, 800 }; VPX_SendCommand("moveWindow PenPlot %d %d %d %d",r.left,r.top,r.right,r.bottom) VPX_SendCommand("moveWindow GazeSpace 15 38 25 73");</pre>	

14.14.4 Specify ViewPoint Window Layout	
GUI:	Windows > { WindowName } (Toggle) Windows > Arrange > { Starup Layout, Cascade }
CLP Command:	<code>setWindow windowNameString windowState</code> <code>setWindow windowLayout</code> windowNameString: Main, EyeCamera, EyeSpace, Controls, Status, GazeSpace, PenPlot, Stimulus windowState: Show, Hide, Maximize, Minimize windowLayout: Startup, Cascade
SDK Function:	<code>int VPX_Video_WindowVisible (BOOL tf);</code> <code>int VPX_StimWind_Hide(void);</code>
<p>Allows the user to easily control the state of all ViewPoint EyeTracker windows.</p> <p>setWindow Startup: equivalent to: Windows > Arrange > Startup layout</p> <p>setWindow Cascade: equivalent to: Windows > Arrange > Cascade</p> <p>Note: “setWindow Stimulus Hide” is different from the command “stimulusGraphicsOptions – Image”, the latter suppresses showing the bitmap image inside the stimulus window.</p> <p>See also:</p> <ul style="list-style-type: none"> <code>moveWindow</code> <code>stimWind_FullDisplay</code> <code>stimWind_AutoShowOnCalibrate</code> <p>Deprecated:</p> <ul style="list-style-type: none"> <code>stimWind_Hide</code> 	
<pre>VPX_Send Command("setWindow Stimulus Show"); VPX_Send Command("setWindow GazeSpace Minimize"); VPX_Send Command("setWindow PenPlot Maximize");</pre>	

14.14.5 Clear Event History window	
GUI:	Windows > Clear History
CLP Command:	<code>eventHistory_Clear</code> No arguments.
SDK Function:	-none-
Clears the EventHistory window.	
<code>VPX_SendCommand("eventHistory_Clear");</code>	

14.14.6 Save window layout settings	
GUI:	File > Settings > Save Window Layout
CLP Command:	settingsFile_SaveWindowLayout No arguments.
SDK Function:	-none-
<p>The window layout information can be saved in a Settings file. It is not saved as part of the standard Save Settings operation.</p> <p>See also:</p> <ul style="list-style-type: none"> settingsFile_Save settingsFile_Load 	
<pre>VPX_SendCommand("settingsFile_SaveWindowLayout");</pre>	

14.15 Settings File commands

14.15.1 Load Settings File	
GUI:	File > Settings > Load Settings ... ^L
CLP Command:	settingsFile_Load <i>filename</i> <i>filename</i> : Name of the settings file to be loaded.
SDK Function:	-none-
<p>Loads a settings file of the specified file name. Recursion is not allowed. Nesting depth is limited to 9.</p> <p>See also:</p> <ul style="list-style-type: none"> settingsFile_SaveWindowLayout 	
<pre>VPX_SendCommand("settingsFile_Load researchsettings");</pre>	

14.15.2 Edit Settings File	
GUI:	File > Settings > Edit Settings File ...
CLP Command:	settingsFile_EditDialog
SDK Function:	-none-
<p>Enables user to quickly access and edit Settings files.</p>	
<pre>VPX_SendCommand("settingsFile_EditDialog");</pre>	

14.15.3 Show Verbose Settings File Loading Details in Event History	
GUI:	File > Settings > Verbose Loading
CLP Command:	settingsFile_Verbose BoolValue <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
Default:	No
Specifies whether to report verbose loading details.	
<code>VPX_SendCommand("settingsFile_verbose Yes");</code>	

14.15.4 Save Settings e.g. calibrations etc.	
GUI:	File > Settings > Save Settings ... Alt-Shift-S
CLP Command:	settingsFile_Save Filename <i>Filename: Name of the settings file to be saved.</i>
SDK Function:	-none-
Saves a settings file with the specified file name.	
See also: settingsFile_SaveWindowLayout	
<code>VPX_SendComand("settingsFile_save researchsettings");</code>	

14.16 SettingsFileList commands

14.16.1 Initialize Settings File List	
GUI:	-none-
CLP Command:	settingsFileList_Init
SDK Function:	-none-
Initializes the list for settings file, making it ready for new settings to be entered.	
<code>VPX_SendCommand("settingsFileList_Init");</code>	

14.16.2 Next Settings File in List		
GUI:	File > settings > SettingsFileList > Next Settings File	F9 (default)
CLP Command:	settingsFileList_Next	
SDK Function:	-none-	
Executes the settings for the next settings file in the settings file list.		
<code>VPX_SendCommand("settingsFileList_Next");</code>		

14.16.3 Add Settings File to the List		
GUI:	-none-	
CLP Command:	settingsFileList_AddName <i>FileName</i> <i>FileName</i>: Name of the settings file to be added to the settings file list.	
SDK Function:	-none-	
Adds a settings file name to the settings file list.		
<code>VPX_SendCommand ("settingsFileList_AddName subject1.txt");</code>		

14.16.4 Restart Settings File List		
GUI:	File > Settings > SettingsFileList > Restart SettingsFileList	
CLP Command:	settingsFileList_Restart	
SDK Function:	-none-	
Reopens the settings file list and re-applies the settings listed in the setting files.		
<code>VPX_SendComand("settingsFileList_Restart");</code>		

14.16.5 Toggle Autosequencer ON / OFF		
GUI:	File > settings > settingsfilelist > Auto Sequencer	F10 (default)
CLP Command:	settingsFileList_AutoSequence <i>BoolValue</i> <i>BoolValue</i>: Yes, No, True, False, On, Off, 1, 0, Toggle	
SDK Function:	-none-	
Enables or disables settings file list sequencer.		
<code>VPX_SendCommand("settingsFileList_AutoSequence on");</code>		

14.16.6 Specify delay between Settings Files in List	
GUI:	-none-
CLP Command:	settingsFileList_SequenceSeconds <i>TimeValue</i> <i>TimeValue: Amount of sequence seconds.</i>
SDK Function:	-none-
<p>Creates a delay between loading settings-files from the settings-file-<i>list</i>. Users may also specify sequence-seconds within the settings files themselves rather than in a settings-file-list.</p> <p>Will create a delay in the execution of settings files from the list for as long as specified. Also note that the sequence seconds can be set within a settings file itself</p>	
<pre>VPX_SendCommand("settingsFileList_SequenceSeconds 4.5");</pre>	

14.17 Torsion commands

14.17.1 Start / Stop Torsion Calculations	
GUI:	Torsion window, Start / Stop button ^T
CLP Command:	torsion_Calculation <i>BoolValue</i> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
Default:	off
<p>Starts and stops the torsion calculations. As a side effect this also adds or removes the torsion stripChart from from the penPlot window.</p> <p>Note 1: In previous PC versions this command would open and close the torsion window as well, because in those previous versions, torsion would be calculated if and only if the torsion window were open.</p> <p><i>See also:</i> VPX_GetTorsion2</p>	
<pre>VPX_SendCommand("torsion_Calculation On");</pre>	

14.17.2 Adjust Start Point of Torsion Sampling Arc	
GUI:	Torsion window, Angle slider
CLP Command:	<code>torsion_SampleAngle <i>FloatDegrees</i></code> <i>FloatDegrees</i> : floating point value 0.0 to 360.0 degrees
SDK Function:	-none-
Default:	180
<p>Adjusts the start point of the torsion sampling arc in degrees. Range from 0 - 360. Zero degrees is at the 3 o'clock position, 90 degrees is at the 6 o'clock position. An interesting demonstration and validation can be obtained on a static image by first unchecking the Auto-set after adjust check box and then moving the Angle slider a few degrees. This moves the start point of the sample vector and so the autocorrelation shows a "torsional" rotation of the same number of degrees.</p>	
<pre>VPX_SendCommand("torsion_SampleAngle 166");</pre>	

14.17.3 Adjust Radius of Torsion Sampling Arc	
GUI:	Torsion window, Radius slider
CLP Command:	<code>torsion_SampleRadius <i>FloatValue</i></code> <i>FloatValue</i> : normalized floating point number 0.01 - 0.99
SDK Function:	-none-
Default:	0.50
<p>Adjust the radius (the distance out from the center of the pupil) of the torsion sampling arc. The arc should be adjusted such that: (a) there is good variation in the striations and marks of the iris, (b) the arc does not include any reflections, such as the glint, that do not move with the iris, (c) the arc does not extend beyond the EyeCamera window.</p>	
<pre>VPX_SendCommand("torsion_SampleRadius 0.75");</pre>	

14.17.4 Autoset Torsion Template after Adjustments	
GUI:	Torsion window, Auto-Set after adjust checkbox
CLP Command:	<code>torsion_AutoSetAfterAdjust</code> <i>BoolValue</i> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
Default:	On
If ON, a new autocorrelation template is set whenever adjustments are made to the radius and angle.	
<code>VPX_SendCommand("torsion_autosetafteradjust On");</code>	

14.17.5 Display Real-Time Torsion Data	
GUI:	Torsion window, Real-time graphics checkbox
CLP Command:	<code>torsion_RealTimeGraphics</code> <i>BoolValue</i> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
Default:	off
Specifies whether to display real-time torsion data in the torsion window. If ON the window is updated every new field. If OFF, the window is updated every 30 th field. This does not affect the real-time data stored in the data file, but does effect the CPU load.	
<code>VPX_SendCommand("torsion_RealTimeGraphics On");</code>	

14.17.6 Adjust Torsion Measurement Range	
GUI:	-none-
CLP Command:	torsion_MeasureDegrees <i>FloatDegrees</i>
SDK Function:	-none-
Default:	+/- 9.0
<p>Adjusts the torsion measurement range. Default is +/- 9 degrees.</p> <p>Since the eye does not normally rotate about the line of sight more than about 9 degrees there is usually no need to perform the auto-correlation past this range, because increasing the range increases the cpu load unnecessarily. There are some situations in which this range needs to be increased, such as when the entire head is rotated. Depending upon the power of your computer, you may need to reduce the resolution of the auto-correlation via: torsion_ResolutionDegrees.</p> <p>See also: torsion_ResolutionDegrees</p>	
<pre>VPX_SendCommand("torsion_MeasureDegrees 9.0");</pre>	

14.17.7 Adjust Torsion Measurement Resolution	
GUI:	-none-
CLP Command:	torsion_ResolutionDegrees <i>FloatDegrees</i> <i>FloatDegrees: floating point value between: 0.20 to 360.0</i>
SDK Function:	-none-
Default	0.5
<p>Adjusts the default torsion measurement resolution. The default is 0.5. This minimum value is 0.20. To limit CPU load, vary this inversely with Torsion_MeasureDegrees.</p> <p>See for further discussion: torsion_MeasureDegrees</p>	
<pre>VPX_SendCommand("torsion_ResolutionDegrees 0.80");</pre>	

14.17.8 Set Autocorrelation Template	
GUI:	Torsion window, Set Template button
CLP Command:	<code>torsion_SetTemplate</code>
SDK Function:	-none-
Sets a new autocorrelation template.	
<code>VPX_SendCommand("torsion_SetTemplate");</code>	

14.18 Interface settings commands

14.18.1 Turn Cursor Control On / Off	
GUI:	Interface > CursorControl > Eye Moves Mouse ^E
CLP Command:	<code>cursor_Control</code> <i>BoolValue</i> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	<code>VPX_CursorControl (bool tf);</code>
Default:	off
Turns Cursor Control feature on or off.	
<code>VPX_SendCommand("Cursor_Control On");</code>	

14.18.2 Use Fixation to Issue Button Click	
GUI:	Interface > CursorControl > Fixation Clicks Buttons ^C
CLP Command:	<code>cursor_DwellClick</code> <i>BoolValue</i> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
Default:	off
Turns on or off dwell time issues mouse click.	
<code>VPX_SendCommand("cursor_DwellClick On");</code>	

14.18.3 Specify Fixation Time to Issue Button Click	
GUI:	Controls window, DataCriteria tab, Mouse Click if Fixated slider
CLP Command:	cursor_DwellSeconds FloatValue <i>FloatValue</i> : From 0.00 to 9.00. Sets the amount of time until mouse click is issued due to fixation.
SDK Function:	-none-
Specifies the dwell time in seconds before a mouse click is issued.	
VPX_SendCommand("cursor_DwellSeconds 2.5");	

14.18.4 Use Blinks to Issue Button Click	
GUI:	Interface > Cursor Control > Blinks Click Buttons (toggle)
CLP Command:	cursorBlinkClick BoolValue <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK Function:	-none-
Default:	Off
Blinks will stimulate mouse clicks with this option enabled.	
VPX_SendCommand("cursorBlinkClick On");	

14.19 RemoteLink & SerialPort controls

14.19.1 Connect / Disconnect Serial Port	
GUI:	Interface > Serial Port > Connect
CLP Command:	serialPortConnect BoolValue <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK Function:	-none-
Default:	Off
Connects and disconnects to the serial port. If disconnected, <i>ViewPoint</i> will not be able to receive a subsequent connect instruction via the serial port.	
VPX_SendCommand("serialPortConnect On");	

14.19.2 Specify Serial Data to Send	
GUI:	Interface > Serial Port > Nothing, Stream, Test, Events, SinglePacket
CLP Command:	<code>serialPortSend Nothing, Stream, Test, Events, SinglePacket</code>
SDK Function:	-none-
Specifies what type of serial data to send.	
<code>VPX_SendCommand("serialPortSend Events");</code>	

14.19.3 Send Serial Port Ping	
GUI:	Interface > Serial Port > Send Ping alt + shift + P
CLP Command:	<code>serialPortPing</code>
SDK Function:	-none-
Sends a PING packet out over the serial connection. The RemoteLink program (or your program) is expected to reply with a PONG packet. When a PONG packet is received, the round trip time is reported.	
<code>VPX_SendCommand("serialPortPing");</code>	

14.20 HeadTracking commands

14.20.1 Connect / Disconnect Head Tracker Interface	
GUI:	HeadTrack > Connect
CLP Command:	<code>headTrackerConnect BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
Default:	Off
Connects or disconnects the head tracker interface. <i>See also:</i> <code>dataFile_AsynchHeadData</code>	
<code>VPX_SendCommand("headTrackerConnect On");</code>	

14.20.2 Specify whether to use local or global origin	
GUI:	HeadTrack > Use Local Origin
CLP Command:	<code>headTracker_useLocalOrigin</code> <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK Function:	-none-
Default:	off
Specifies whether the head sensor position and angle data should be relative to the transmitter, or relative to a local origin set by other commands. See also: <code>headTracker_setLocalOrigin</code> <code>headTracker_resetLocalOrigin</code>	
<code>VPX_SendCommand("headTracker_useLocalOrigin On");</code>	

14.20.3 Reset Origin for the Head Sensor	
GUI:	HeadTrack > Reset Local Origin
CLP Command:	<code>headTracker_resetLocalOrigin</code>
SDK Function:	-none-
Sets the current location of the head sensor as the local origin (for the six degree of freedom, position and angle). All subsequent position and angle data are relative to this new origin.	
<code>VPX_SendCommand("headTracker_resetLocalOrigin");</code>	

14.20.4 Set Position and Angle Origins	
Control or Menu:	-none-
CLP Command:	<code>headTracker_setLocalOrigin</code> <i>coordinatesAndAngleValues</i> <i>coordinatesAndAngleValues</i> : x y z roll pitch yaw
SDK Function:	-none-
Sets the local origin (position and angle). All subsequent position and angle data are relative to this new origin.	
<code>VPX_headtracker_setLocalOrigin</code>	

14.20.5 Specify the Vector between Head Sensor and the Eyeball.	
Control or Menu:	-none-
CLP Command:	headTracker_eyeOffset <i>Coordinates</i> <i>Coordinates: x y z</i>
SDK Function:	-none-
Specifies the vector offset between the head sensor and the eyeball. This information is used by ViewPoint for correctly displaying the intercept point of the primary axis of the eye when the headspace window is enabled.	
<code>VPX_SendCommand("headTracker_eyeOffset 0.7 0.5 0.3");</code>	

14.20.6 Turn CRT pulse synchronization On / Off	
GUI:	HeadTrack > CRT Sync > * Off, 50-72 Hz, 73-144 Hz
CLP Command:	headTracker_CRT_sync <i>Off, Low, High</i>
SDK Function:	-none-
CRT monitor emits varying magnetic fields that cause errors with magnetic trackers. You can compensate for this by synchronizing the FOB to the monitor using the CRT SYNC Pickup. Low 50- 72 Hz and high is 73-144 Hz.	
<code>VPX_SendCommand("headTracker_CRT_sync Low");</code>	

14.21 Binocular commands

14.21.1 Turn Binocular Mode On / Off	
GUI:	Binocular > Binocular mode (toggle)
CLP Command:	binocular_Mode <i>BoolValue</i> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK Function:	-none-
Default:	Off
Turns binocular operation mode on or off. When On, the data file will automatically include the data from Eye_B and this data will be available in real-time via the SDK.	
<code>VPX_SendCommand("Binocular_Mode On");</code>	

14.21.2 Specifies Binocular Averaging	
GUI:	Binocular > Show both eye positions Show average Y-gaze positions Show average of eye positions
CLP Command:	binocular_Averaging <i>averageOption</i> <i>averageOption</i> : Off, only_Y, both_XY
SDK Function:	-none-
Specifies whether to use binocular averaging and which type.	
<pre>VPX_SendCommand("binocular_Averaging both_XY ");</pre>	

14.22 File Related

14.22.1 Launch ViewPoint with Command Line Options	
GUI:	-none-
CLP Command:	launchApp <i>applicationName</i> <i>argumentsToApp</i> <i>applicationName</i> : The name of the application to launch <i>argumentsToApp</i> : command line arguments may be flags, input/output file names, etc.
SDK Function:	VPX_LaunchApp (<i>applicationName</i> , <i>argsToApp</i>);
<p>A general purpose command to launch an application and to provide it with optional command line arguments.</p> <p>Note: Must use full path for file names if they are not in the default location for the application.</p> <p>Special: This command will enable playing stimulus movies.</p> <p>WARNING: The CLP strings are parsed by the ViewPoint application, therefore you cannot use this CLP string to launch the ViewPoint application itself, instead use: VPX_LaunchApp("ViewPoint.exe","");</p> <p>See also: VPX_QuitViewPoint ();</p>	
<pre>launchApp DataMarker.exe launchApp "VP_MoviePlayer.exe" "C:/ARI/Movies/m 1.mov" VPX_SendCommand ("launchApp VP_MoviePlayer.exe \"C:/ARI/Movies/m 6.mov\" "); VPX_LaunchApp("ViewPoint.exe", " "); // VPX_SendCommand ("launchApp ViewPoint.exe"); // PARSE ERROR IF VP NOT RUNNING</pre>	

14.22.2 Specify Default ViewPoint Folder path	
GUI:	-none-
CLP Command:	<pre>setPath pathID pathString pathID: IMAGES: DATA: SETTINGS: SOUNDS: DOCUMENTATION: pathString : full path string or keyterm: DEFAULT_PATH</pre>
SDK Function:	-none-
<p>Specifies the current default folder path for important ViewPoint directories.</p> <p>CAUTION: the current version requires that (a) only forward slashes (/) be used, and (b) a final forward slash be used; otherwise bad things may happen.</p> <p>The special keyTerm DEFAULT_PATH may be used to reset the path to the original ViewPoint default path string.</p> <p>The key term IMAGES: must include the colon at the end.</p> <p>See also:</p> <pre>PTCHAR VPX_GetViewPointHomeFolder(PTCHAR pathString);</pre>	
<pre>setPath IMAGES: DEFAULT_PATH setPath IMAGES: "C:/ARI/Global/Images/"</pre>	

14.23 FKey

14.23.1 Associate CLP Commands with FKeys	
GUI:	-none-
CLP Command:	fkey_cmd fKeyNumber commandString fKeyNumber : integer in { 1 to 12 } commandString : any valid command
SDK Function:	-none-
<p>Allows CLP commands to be associated with FKeys. It is a very useful to customize the Fkeys for your needs.</p> <p>These fKey associations can be viewed in the Info panel: menu Help > Info > ShortCuts tab.</p> <p><i>Note:</i> do not put the command string in quotes; all text after the number is included in the command string including leading spaces and tabs, UNLESS the string contains a semicolon, which has higher priority than the Fkey_cmd command does, so anything after the semicolon will be processed independently after the Fkey_cmd command is processed. <i>Important:</i> this is subject to change.</p> <p><i>Restore defaults with:</i> fkey_default</p>	
<pre>VPX_SendCommand("fkey_cmd 12 dataFile_NewUnique"); VPX_SendCommand("fkey_cmd 11 stimulus_playSoundFile \"No Way .wav\" ");</pre>	

14.24 TTL

14.24.1 Associate CLP Commands with TTL Voltage Changes	
GUI:	-none-
CLP Command:	ttl_cmd signedChannel commandString signedChannel : +/- just before an integer in { 0 to 7 } commandString : any valid command
SDK Function:	-none-
<p>Allows CLP commands to be associated with TTL voltage changes, high or low.</p> <p>These associations can be viewed in the Info panel: menu Help > Info > TTL CMDS tab.</p> <p><i>Restore defaults with:</i> ttl_default</p> <p>Requires the TTL option.</p>	
<pre>VPX_SendCommand("ttl_cmd +0 dataFile_Pause"); VPX_SendCommand("ttl_cmd -0 dataFile_Resume"); VPX_SendCommand("ttl_cmd +1 dataFile_NewUnique"); VPX_SendCommand("ttl_cmd -1 historyReport \"TTL ch#1 LO\" "); VPX_SendCommand("ttl_cmd +2 dataFile_InsertMarker 2");</pre>	

14.24.2 Set TTL Output Voltages	
GUI:	-none-
CLP Command:	<code>t11_out signedChannel</code> <code>signedChannel : +/- just before an integer in { 0 to 7 }</code>
SDK Function:	-none-
<p>Provides a mechanism to easily set the TTL output voltages. There must be no space between the sign and the number. Requires the TTL option.</p>	
<pre>VPX_SendCommand("t11_out -0"); // set TTL channel 0 LO VPX_SendCommand("t11_out +7"); // set TTL channel 7 HI</pre>	

14.24.3 Simulate Change in TTL Input	
GUI:	-none-
CLP Command:	<code>t11_simulate signedChannel</code> <code>signedChannel : +/- just before an integer in { 0 to 7 }</code>
SDK Function:	-none-
<p>A change in TTL input can be simulated to the software to aid in development and debugging. Does not require the TTL option.</p>	
<pre>VPX_SendCommand("t11_simulate +0"); // simulate TTL channel 0 HI event</pre>	

14.24.4 Set TTL Output to Indicate Data Quality Codes

GUI: `-none-`

CLP Command: `ttl_out_quality channel levelString`
`channel` : integer in { 0 to 7 }
`levelString` : OFF, or any of VPX_QUALITY_*
 VPX_QUALITY_PupilScanFailed
 VPX_QUALITY_PupilFitFailed
 VPX_QUALITY_PupilCriteriaFailed
 VPX_QUALITY_PupilFallBack
 VPX_QUALITY_PupilOnlyIsGood
 VPX_QUALITY_GlintIsGood

SDK Function: `-none-`

A ttl output channel can be specified to indicate when the data quality value is **greater than or equal to** (`>=`) the quality criterion level.

The level strings are identical to the VPX_QUALITY_* constants defined in the VPX.h file. The best quality is level == QUALITY_GlintIsGood, poorer quality raises the quality level. Setting the quality criterion to x will cause the TTL channel to always be high, because the data quality value is always greater than or equal to this.

See also:

`VPX_GetQuality`
`verbose +ttl_out`

```
VPX_SendCommand( "ttl_out_quality 0 VPX_QUALITY_PupilFallBack" );
```


14.25 Misc

14.25.1 Specify Verbose Information to Send to History Window	
GUI:	-none-
CLP Command:	verbose +/-activityType activityType : ttl_out ttl_cmd calibration settings
SDK Function:	-none-
<p>CURRENTLY BEING EXPANDED</p> <p>Allows fine control over the type of verbose information sent to the History window. The reports are turned on or off by preceding the keyTerm with a plus (+) or minus(-), respectively. There can be no space between the sign and the keyTerm. This is used for debugging and the details of what is reported may change without notice. The following arguments can be used to turn reporting on or off.</p> <pre>+setting +parsing +ttl_out +ttl_cmd +frameGrabber +videoTiming +serialSend +serialReceive; +insertMark +insertString +insertUserTag +calibration +headTracker +nameList</pre>	
<pre>VPX_SendCommand ("verbose +ttl_cmd +ttl_out -calibration -settings");</pre>	

14.25.2 Update Eye Data on Request	
GUI:	Alt-Shift-U
CLP Command:	updateData
SDK Function:	-none-
<p>Some programs want as much CPU time as they can get, so they would like to have ViewPoint video image processing turned off until fresh data is needed. We have now added the capability to update eye data based on the most recent video image in memory (this memory is constantly being updated via direct memory access, DMA, by the video capture board).</p> <p><i>Note:</i> sending an updateData command while NOT frozen may cause a glitch in the data timing.</p>	
<pre>VPX_SendCommand ("updateData");</pre>	

14.25.3 Set Status Window Update Rate for FPS Field	
GUI:	-none-
CLP Command:	fpsUpdate nth_Interrupt nth_Interrupt : integer
SDK Function:	-none-
<p>Added control for rate of update of the FPS (frames per second) value in the Status window. The argument may be any positive number. An argument of 1 would cause the fps calculation to be updated every video interrupt (frame or field), 2 would be every 2nd, etc.</p>	
<pre>VPX_SendCommand ("fpsUpdate 3");</pre>	

14.25.4 SDK Debug Mode	
GUI:	-none-
CLP Command:	debugSDK
SDK Function:	VPX_DebugSDK(int onOff)
<p>This command will add debugging capability for the dll based sdk.</p>	
<pre>VPX_DebugSDK(1);</pre>	

14.25.5 Specify ViewPoint Generated Events	
GUI:	-none-
CLP Command:	<code>vpX_event +option -option</code> <code>option : videoSynch</code>
SDK Function:	-none-
<p>This provides a mechanism to control (thin) the number of events that <i>ViewPoint</i> generates. The command format the keyword <code>vpX_event</code> followed by one or more options that are immediately preceded (no spaces) by a + or - character.</p> <p><i>Note:</i> unnecessary messages will unnecessarily consume system resources.</p> <p>See: <code>VPX_VIDEO_SynchSignal</code></p>	
<code>VPX_DebugSDK(1);</code>	

14.25.6 Turn Accelerator Key Functionality On / Off	
GUI:	-none-
CLP Command:	<code>acceleratorKeys BoolValue</code> <code>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</code>
SDK Function:	<code>VPX_AcceleratorKeys (bool tf);</code>
Default:	On
<p>RARE</p> <p>Specifies whether accelerator keys may be used.</p>	
<code>VPX_SendCommand("acceleratorKeys Off");</code>	

14.26 Parser Instructions

14.26.1 Settings File Comment	
GUI:	-none-
CLP Command:	<code>COMMENT</code> <code>//</code>
SDK Function:	-none-
<p>For use in a settings file. Use either the key word <code>COMMENT</code> or the double forward slashes (<code>//</code>) to tell the parser to ignore this line.</p>	
<code>//</code>	

14.26.2 End of Settings File Command	
GUI:	-none-
CLP Command:	END
SDK Function:	-none-
The command END should be placed by itself on the final line of a settings file. This provides a way to verify that all the settings command lines were read. <i>ViewPoint</i> reports "Settings read successfully" when the END command is reached.	
END	

Chapter 15 Software Developers Kit (SDK)

15.1 General

A third party application (for example, your application) may interact with the *ViewPoint EyeTracker*® by compiling with the `VPX_InterApp.lib` file, a library file. At run time your program will dynamically link to the `VPX_InterApp.DLL`, a dynamic-link library file. All of the *ViewPoint* inter-application communications constants, data types, and functions begin with the prefix `VPX_` and are specified in the `VPX.h` file.

Your application may call the `VPX_` routines `VPX_LaunchApp("ViewPoint.ext", " ")` and `VPX_QuitViewPoint()` to control when the *ViewPoint EyeTracker*®, application is running. If the remote application will provide its own control settings to *ViewPoint*, then it may be desirable to launch only the [EyeCamera](#) window, by calling:

```
VPX_LaunchApp("ViewPoint.ext", " ")
VPX_LaunchApp("ViewPoint.ext", " -hideMain -freeEyeCamera ")
```

The command line argument `-minimized` allows access to the main *ViewPoint* program window via the minimized icon, `-hideMain` makes the main window of *ViewPoint* completely inaccessible. `-freeEyeCamera` launches *ViewPoint* with the eye camera window as a free floating window.

Note: currently, if launched minimized, the splash window is not presented.

Your application may access *ViewPoint* data at any time. It should be noted that employing a tight polling loop is a poor programming practice, because it unnecessarily consumes computer CPU time. If only occasional updates are required, a timer would probably be the preferred method (see `MSWindows SetTimer` function). If your application needs to know immediately every time the data values are updated, then it should register to receive notifications of fresh data.

15.2 Registering to Receive Notifications

Registering to receive notifications of fresh data is a multi-step process:

Obtain the unique message identifier used by *ViewPoint* for inter-process communication:

```
const static UINT wm_VPX_message = RegisterWindowMessage(VPX_MESSAGE).
```

The window(s) that wish to receive notification must register to receive it by calling:

```
VPX_InsertMessageRequest( m_hWnd, wm_VPX_message ).
```

More than one window in an application may request notifications, however no window should make more than one request.

Your program must listen for the notifications. This may be done in several ways. If you are using MFC message maps, then you should add a mapping, e.g.:

```
ON_REGISTERED_MESSAGE( wm_VPX_message, OnVPX_message ).
```

If you are using Win32, then you will want to listen for the `wm_VPX_message` notification just as you would listen for a `WM_PAINT` message.

When a `wm_VPX_message` is received, your message handling routine should determine the type of the notification, as follows:

```
WORD notificationCode = HIWORD( wParam ).
```

This `notificationCode` may then be used in a switch statement that uses case constants defined in the `VPX.h` file. Among others, these include: `VPX_DAT_FRESH`, `VPX_ROI_CHANGE`, and notifications relating to the presentation of calibration point stimuli. Refer to Table 40 for a complete list.

IMPORTANT: A finite number of message windows may register a `VPX_InsertMessageRequest` at one time (currently ten), after which notification requests will be refused. You application should always make sure that each requesting window successfully removes its request for notification before the window is destroyed or before the program terminates, by calling: `VPX_RemoveMessageRequest(m_hWnd)`.

The *ViewPoint EyeTracker*[®] **Status** window contains the field **DLL sharing**. This shows the number of windows that are currently registered to receive notifications. You may monitor this value during program development to make sure that terminating your application also decrements this value. If you find that all requests are used up, the only way to reset this list is to terminate every application that dynamically links to the `VPX_InterApp.DLL`.

Note 1: The `VPX_Set_x` routines work by directly sending values to the *ViewPoint EyeTracker*[®]. Any commands sent before *ViewPoint* is launched, will have no effect on the initial startup values of *ViewPoint*.

Note 2: *ViewPoint* does not send out notifications when control values are changed, so unless the remote application explicitly sets the *ViewPoint* values, the remote applications control values (e.g. of sliders) will not accurately reflect the *ViewPoint EyeTracker*[®] application control values.

We encourage users and third party developers to work with us in developing this interface. We take suggestions and usability reports very seriously.

15.3 Example SDK Code

Your application must first register with the `VPX_InterApp.lib` by doing the following for each window procedure (WinProc) that desires notification:

```
if ( uniqueMessageId == 0 ) {
    uniqueMessageId = RegisterWindowMessage( VPX_MESSAGE );
    VPX_InsertMessageRequest( hWnd, uniqueMessageId );
}
```

After registering, the WinProc should listen for notifications:

```
LRESULT CALLBACK WndProc ( HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam )
{
    WORD notificationCode = HIWORD(wParam) ;
    if ( message == uniqueMessageId ) {
        switch ( notificationCode )
        {
            case VPX_DAT_FRESH :
                {
                    EyeType eye = LOWORD ( wParam );
                    showFreshData( eye graphicsWindow );
                }
            } break;
        }
    }
}
```

15.4 Data Quality Codes

Because the data can now be from multiple sources, and because in data files it is often desirable to store all data on a single record line, the concept of a data reject record is obsolete. Instead, data quality codes are provided for each source.

A `VPX_DAT_FRESH` is sent and the SDK application is responsible for obtaining the quality code and making a decision as to what levels of quality are appropriate for various situations. We highly recommend that you use the constants provided, rather than testing the integer value, as these are not guaranteed to remain unchanged. Nevertheless, the hierarchical relationships are expected to remain intact. For example:

```
VPX_GetDataQuality2( eye, &quality );
switch (quality)
{
    case VPX_QUALITY_PupilScanFailed: showGaze(false); break;
    case VPX_QUALITY_PupilFitFailed: showGaze(false); break;
    case VPX_QUALITY_PupilCriteriaFailed: showGaze(false); break;
    case VPX_QUALITY_PupilFallBack: showGaze(true ); break;
    case VPX_QUALITY_PupilOnlyIsGood: showGaze(true ); break;
    case VPX_QUALITY_GlintIsGood: showGaze(true ); break;
}
```

Data records are constantly stored, because the data quality may vary with different sources. For example, data may be good from one eye, and a wink occurred in the other eye, or both eyes are closed, but the head tracker data is still good. Data sections now contain individual quality columns, as needed.

15.5 Sending CLP Commands with the SDK

The *ViewPoint EyeTracker*[®] software developers kit (SDK) provides a routine that allows sending Command Line Parser (CLP) strings directly to the *ViewPoint EyeTracker*[®]. This means that the same command strings that are loaded from Settings Files or that are sent in serial port packets, can be issued via the DLL. An important benefit is that this allows passing textStrings and fileNames.

```
result = VPX_SendCommand( TCHAR *cmd );
VPX_SendCommand("dataFile_NewName rabbitPictureData");
VPX_SendCommand("dataFile_InsertString Showing Picture of a Rabbit");
VPX_SendCommand("settingsFile_Load RabbitPictureROI.txt");
```

The function returns an integer result value that provides feedback about the success or problems encountered. See `vpX.h` for a list of return codes.

For a matrix comparing CLP commands, SDK functions, window buttons, and menu items, refer to the Command and Control Matrix contained in the SDK folder.

15.6 High Precision Timing

High Precision timing (HPT) with resolution in the order of 0.0000025, i.e., 2.5E-6 or 2.5 microseconds, is available via the SDK function call:

```
#define SINCE_PRECISE_INIT_TIME ((double*)NULL)
#define RESET_PRECISE_HOLD_TIME 1
#define LEAVE_PRECISE_HOLD_TIME 0

double seconds = VPX_GetPrecisionDeltaTime(double *holdTime, BOOL resetHoldTime);
```

The function returns the difference between the holdTime and the current time. When NULL is passed to `VPX_GetPrecisionDeltaTime` it returns the time since the DLL was first initialized, which is when it was first called. If `resetHoldTime` is true, the variable `holdTime` will be set to the current time when the function returns.

15.7 DLL Version Checking

All applications using the ViewPoint SDK should include the following check for a possible mismatch between the version of the DLL (loaded at runtime) and the version of the SDK library (prototypes and constants) that was compiled into the application.

```
BOOL versionMismatch = VPX_VersionMismatch (VPX_SDK_VERSION);

double dllVersion = VPX_GetDLLVersion();
if ( VPX_SDK_VERSION != dllVersion ) doSomething();
```

15.8 SDK Access Functions

There are many functions provided to access ViewPoint data and the current state of the

ViewPoint Program. These are generally referred to as the **Get** commands and **Set** commands.

15.8.1 Get Eye Data Access

```
int ok = VPX_GetGazePoint ( VPX_RealPoint* gazePoint )
int ok = VPX_GetGazePoint2 ( VPX_EyeType eye, VPX_RealPoint* gazePoint )
int ok = VPX_GetGazePointSmoothed2( VPX_EyeType eye, VPX_RealPoint* gazePnt );
```

Retrieves the calculated position of gaze, for either Eye_A or Eye_B if binocular mode is on. Monocular ViewPoint uses Eye_A by default. Returns 1 if valid VPX_EyeType, 0 otherwise.

See also:

```
VPX_GetPupilPoint2
VPX_GetGlintPoint2
```

```
VPX_RealPoint rp ;
VPX_GetGazePoint2( EYE_A, &rp );
printf( " X: %g , Y: %g ", rp.x, rp.y );
```

```
int VPX_GetGazeAngle2( VPX_EyeType eye, VPX_RealPoint *gp )
int VPX_GetGazeAngleSmoothed2( VPX_EyeType eye, VPX_RealPoint *gp )
```

Retrieves the calculated angle of gaze, for either Eye_A, or Eye_B if binocular mode is on. Monocular ViewPoint uses Eye_A by default.

Note: The angles are from trigonometric calculations based on the values that the user has measured and set for the window size (horizontal & vertical) and the viewing distance. This is **NOT** for use with the head tracker option.

See also:

```
VPX_GetMeasuredScreenSize
VPX_GetMeasuredViewingDistance
VPX_GetHeadPositionAngle
```

```
VPX_RealPoint rp ;
VPX_GetGazeAngleSmoothed2( EYE_A, &rp );
printf( " X: %g , Y: %g ", rp.x, rp.y );
```

```
int ok = VPX_GetFixationSeconds ( double* seconds )
int ok = VPX_GetFixationSeconds2 ( VPX_EyeType eye, double* seconds )
```

Retrieves the number of seconds that the total velocity has been below the VelocityCriterion for either Eye_A or Eye_B if binocular mode is on. Monocular ViewPoint uses Eye_A by default. .

A zero value indicates a saccade is occurring.

Returns 1 if valid VPX_EyeType, 0 otherwise.

This function replaces the less precise function: VPX_GetFixationDuration(DWORD);

See also:

[VPX_GetTotalVelocity](#)

```
double seconds, milliseconds, microseconds ;
VPX_GetFixationSeconds2 ( EYE_A, &seconds ) ;
milliseconds = 1000.0 * seconds ;
microseconds = 1000.0 * milliseconds ;
```

```
VPX_GetTotalVelocity ( double *velocity);
VPX_GetTotalVelocity2 ( VPX_EyeType eye, double* velocity );
```

Retrieves the total velocity of movement in the (x,y) plane. That is, the first derivative of the (smoothed) position of gaze for either Eye_A, or Eye_B if binocular mode is on. Monocular ViewPoint uses Eye_A by default.

Returns 1 if valid VPX_EyeType, 0 otherwise.

See also:

[VPX_GetComponentVelocity](#)

[VPX_GetFixationSeconds](#)

```
VPX_RealType velocity ;
VPX_VPX_GetTotalVelocity2 ( EYE_A, &velocity );
printf( " Velocity: %g ", velocity );
```

```
VPX_GetComponentVelocity( VPX_RealPoint *velocityComponents );
VPX_GetComponentVelocity2( VPX_EyeType eye, VPX_RealPoint *velocityComponents);
```

Retrieves the x- and y-components of the eye movement velocity for either Eye_A, or Eye_B if binocular mode is on. Monocular ViewPoint uses Eye_A by default.

Returns 1 if valid VPX_EyeType, 0 otherwise.

See also:

[VPX_GetTotalVelocity](#)

```
VPX_RealPoint cv ;
VPX_GetComponentVelocity2( Eye_A, &cv );
printf( " dx/dt: %g , dy/dt: %g ", cv.x, cv.y );
```

```
VPX_GetPupilSize ( VPX_RealPoint *dims )
VPX_GetPupilSize2 ( VPX_EyeType eye, VPX_RealPoint *dims )
```

Retrieves the raw size of the oval fit to the pupil for either Eye_A or Eye_B if binocular mode is on. Monocular ViewPoint uses Eye_A by default.

The x- y-size values are normalized with respect to the EyeSpace dimensions that have a 4:3 aspect, so the x- and y-values are anisotropic. To obtain the aspect ratio of the pupil, rescale: (aspect = ps.x / (ps.y * 0.75)

Returns 1 if valid VPX_EyeType, 0 otherwise.

See also:

[VPX_GetPupilAspectRatio](#)

[VPX_GetPupilOvalRect](#)

```
VPX_RealPoint pupilDimensions ;
double pupilAspectRatioCalculated, pupilAspectRatioRetrieved ;
VPX_GetPupilSize( &pupilDimensions );
// NOTE: dimensions are normalized but incommensurate
pupilAspectRatioCalculated = pupilDimensions.x / (pupilDimensions.y * 0.75 );
VPX_GetPupilAspectRatio( &pupilAspectRatioRetrieved );
difference = pupilAspectRatioCalculated - pupilAspectRatioRetrieved ;
printf( "pupilAspectRatio %g - %g = %g",
    pupilAspectRatioCalculated, pupilAspectRatioRetrieved, difference );
```

```
int VPX_GetPupilAspectRatio( double *ar );
int VPX_GetPupilAspectRatio2( VPX_EyeType eye, double *ar );
```

Retrieves the dimensionless value of pupil aspect ratio. This ratio value is independent of the EyeCamera window shape. A circular pupil will produce a value of 1.0

Retrieves the pupil aspect ratio.

See also:

```
VPX_GetPupilSize
VPX_GetPupilOvalRect
```

See example under: [VPX_GetPupilSize](#)

```
VPX_GetPupilOvalRect ( VPX_RealRect *ovalRect )
VPX_GetPupilOvalRect2 ( VPX_EyeType eye, VPX_RealRect *ovalRect )
```

Retrieves the rectangle in **RAW** EyeSpace coordinates (normalized with respect to the EyeCamera window) that specifies the oval (ellipse) fit to the pupil.

Separate rectangles are available for Eye_A, or Eye_B if binocular mode is on. Monocular ViewPoint uses Eye_A by default.

See also:

```
VPX_GetPupilSize
VPX_RealRect2WindowRECT
VPX_GetPupilAspectRatio
VPX_RealRect2WindowRECT
```

```
// Remote painting of the pupil size and location.
HDC hDC = GetDC( hWnd );
RECT wr, scaledPupilRECT ;
VPX_RealRect pr ;
VPX_GetPupilOvalRect( &pr );
GetClientRect( hWnd, &cr );
VPX_RealRect2WindowRECT( pr, cr, &scaledPupilRECT );
Rectangle( hDC, scaledPupilRECT );
ReleaseDC( hWnd, hDC );
```

```
VPX_GetPupilPoint ( VPX_RealPoint *rawPupilLoc )
VPX_GetPupilPoint2 (VPX_EyeType eye, VPX_RealPoint *rawPupilLoc
```

Retrieves the **raw** normalized (x,y) location of the center of the pupil (center of the oval fit to the pupil) in the EyeSpace, for either Eye_A or Eye_B if binocular mode is on. Monocular ViewPoint uses Eye_A by default. (c.f. VPX_GetPupilCentroid2)

```
VPX_GetPupilPoint2( EYE_A, &rawPupilLocation );
```

```
VPX_GetGlintPoint ( VPX_RealPoint *rawGlintLoc )
VPX_GetGlintPoint2 ( VPX_EyeType eye, VPX_RealPoint *rawGlintLoc )
```

Retrieves the **raw** normalized (x,y) location of the center of the glint (center of the oval fit to the glint) in the EyeSpace, for either Eye_A or Eye_B if binocular mode is on. Monocular ViewPoint uses Eye_A by default. (c.f. VPX_GetGlintCentroid2)

```
VPX_GetGlintPoint2( EYE_A, &rawGlintLocation );
```

```
int VPX_GetDiffVector( VPX_RealPoint *dv)
int VPX_GetDiffVector2 ( VPX_EyeType eye, VPX_RealPoint *dv )
```

Retrieves the **raw** normalized vector difference between the centers of the pupil and the glint in the EyeSpace, for either Eye_A or Eye_B if binocular mode is on. Monocular ViewPoint uses Eye_A by default.

```
VPX_GetDiffVector2( EYE_A, &differenceVector );
```

```
VPX_GetPupilCentroid2 ( VPX_EyeType eye, VPX_RealPoint *centroid
```

Provides **raw** data access to the normalized centroid of the pupil threshold scan in the EyeSpace window, regardless of what subsequent processing options are selected. (c.f. VPX_GetPupilPoint2)

```
VPX_GetPupilCentroid2( EYE_A, &centroid );
```

```
VPX_GetGlintCentroid2 ( VPX_EyeType eye, VPX_RealPoint *gc )
```

Provides raw data access to the normalized centroid of the glint threshold scan in the [EyeSpace](#) window, regardless of what subsequent processing options are selected. (c.f.

```
VPX_GetGlintPoint2 )
```

```
VPX_GetGlintCentroid2 ( EYE_A, &gc );
```

```
VPX_GetTorsion( double *degrees )
```

```
VPX_GetTorsion2 ( VPX_EyeType eye, double *degrees )
```

Retrieves torsion in degrees for either Eye_A or Eye_B if binocularmode is on. Monocular ViewPoint uses Eye_A by default. The ViewPoint torsion measurement must be turned on, it is off by default to save processing time.

```
VPX_GetTorsion2 ( EYE_A, &degrees );
```

```
VPX_GetDataQuality2 ( VPX_EyeType eye, int *quality )
```

Retrieves the quality code for the eye data.
See [VPX.h](#) for a list of data quality constants.

```
VPX_GetDataQuality2 ( EYE_A, &quality );
```

15.8.2 Get Time Information

```
VPX_GetDataTime2 ( VPX_EyeType eye, double *dataTime )
```

Retrieves the high precision time, in seconds, that the video frame became available for the current data point, before video image processing and other calculations were done. This was modified in version 2.8.2.36, Previously this obtained the time that the data was stored to the DLL and a VPX_DAT_FRESH event was issued. Now this function obtains the video sych time that better reflects the actual time that the image of the eye became available, and is not affected by variance in image processing time. The data storage time can now be obtained via VPX_GetStoreTime2.

Note: this modification effects VPX_GetDataDeltaTime2 such that its variance should significantly reduced.

See also:

[VPX_GetStoreTime2](#)

```
VPX_GetDataTime2 ( EYE_A, &dataTime );
```

```
VPX_GetDataDeltaTime2 ( VPX_EyeType eye, double* deltaTime )
```

Retrieves the high precision time interval, in seconds between the last two *dataTime* values.

```
VPX_GetDataDeltaTime2 ( EYE_A, &deltaTime );
```

```
VPX_GetStoreTime2 (VPX_EyeType eye, double* storeTime )
```

Retrieves the high precision timestamp in seconds of the last time the data was stored to the DLL and a VPX_DAT_FRESH event was issued. Use VPX_GetDataTime for eye movement times; use this to find out when the data was available in the DLL, e.g., to calculate the delay in inter-application notification event handling.

```
VPX_GetStoreTime2 ( EYE_A, &deltaTime );
```

15.8.3 Get Motor Data

```
VPX_GetHeadPositionAngle ( VPX_PositionAngle *hpa)
```

Retrieves head position and angle data. The PositionAngle structure is defined in `VPX.h`.
**** available with head tracker option only ****

NOTE: Subject to change.

```
VPX_PositionAngle hpa;  
VPX_GetHeadPositionAngle ( &headTrackerSixDOF );
```

```
POINT VPX_GetCursorPosition();
```

Returns the integer screen coordinates of the mouse cursor.

See also:

cursor_Control

VPX_GetMeasuredScreenSize

```
//Convert from pixel position to normalized screen position.  
POINT pixelCursorPosition = VPX_GetCursorPosition();  
VPX_RealPoint screenSize, normalizeCursorPosition;  
VPX_GetMeasuredScreenSize ( &screenSize );  
normalizeCursorPosition.x = (float) pixelCursorPosition.x / screenSize.x ;  
normalizeCursorPosition.y = (float) pixelCursorPosition.y / screenSize.y ;
```


15.8.4 Get ViewPoint Status

```
int VPX_GetStatus( VPX_StatusItem );
```

This provides a simple way to determine the current status or state of various ViewPoint operations.

Note: The return value may need to be type cast for correct interpretation.

See also:

```
status_Dump
VPX_STATUS_CHANGE
```

```
int running = VPX_GetStatus( VPX_STATUS_ViewPointIsRunning );
// regardless of whether ViewPoint or RemoteLink is the local distributor.
int frozen = VPX_GetStatus( VPX_STATUS_VideoIsFrozen );
int open = VPX_GetStatus( VPX_STATUS_DataFileIsOpen );
int paused = VPX_GetStatus( VPX_STATUS_DataFileIsPaused );
int thresh = VPX_GetStatus( VPX_STATUS_AutoThresholdInProgress );
int calib = VPX_GetStatus( VPX_STATUS_CalibrationInProgress );
int binoc = VPX_GetStatus( VPX_STATUS_BinocularModeActive );
int scene = VPX_GetStatus( VPX_STATUS_SceneVideoActive );
char shape = (char)VPX_GetStatus( VPX_STATUS_StimulusImageShape );
// returns 'I'=isotropic stretch, 'C'=centered, 'F'=fit to window, 'A'=actual
Int dllDataSource = VPX_GetStatus( VPX_STATUS_DistributorAttached );
// returns: 0=VPX_Distributor_None, 1=VPX_Distributor_IsViewPoint, or
//          2=VPX_Distributor_IsRemoteLink
```

```
PTCHAR VPX_GetViewPointHomeFolder( PTCHAR pathString );
```

Concatenates the full path to the *ViewPoint* folder onto the end of the provided string.

Note: this is not a copy operation, i.e., it does not clear any existing contents of the provided string.

To effectively obtain a copy operation, make sure an empty string is provided.

```
TCSTR pictureFile = TEXT(""); // clear VERY IMPORTANT TO DO
VPX_GetViewPointHomeFolder( pictureFile ); // adds: ...ViewPoint/"
lstrcat( pictureFile, IMAGE_FOLDER ); // adds "/Images/"
lstrcat( pictureFile, myPicture.bmp ); // adds "myPicture.bmp"
```

15.8.5 Get ViewPoint Stimulus Window

```
HWND VPX_GetViewPointStimulusWindow(void);
HWND VPX_GetViewPointGazeSpaceWindow(void);
```

Allows a layered program to access to ViewPoint's Stimulus_Window by way of the window handle.

Note: this will work only if the the local data distributor application is *ViewPoint*; it will not work if the local distributor application is *RemoteLink*.

See also:

```
VPX_SetEyeImageWindow
VPX_SetExternalStimulusWindow
VPX_GetStatus( VPX_STATUS_DistributorAttached );
```

```
int dllSource = VPX_GetStatus( VPX_STATUS_DistributorAttached );
if ( VPX_Distributor_IsViewPoint == dllSource ) {
for ( int ix=0; ix<2; ix++ )
{
    HWND hWnd ;
    if ( ix == 0 ) hWnd = VPX_GetViewPointStimulusWindow();
    else hWnd = VPX_GetViewPointGazeSpaceWindow();
    if (!hWnd) continue ;
    CWnd* w = FromHandle( hWnd );
    CDC* d = w->GetDC();
    RECT r ;
    w->GetClientRect(&r);
    d->Rectangle( r.left+10, r.top+10, r.right-10, r.bottom-10 );
    d->MoveTo( r.left, r.bottom ); d->LineTo( r.right, r.top );
    r.left = r.top = 25 ;
    d->DrawText("Drawing from\nVPX_mfc_demo",&r,0);
    ReleaseDC(d);
}
}
```

15.8.6 Get Stimulus Display Geometry

```
VPX_GetMeasuredViewingDistance( VPX_RealType *vd );
```

Provides the physical distance (Millimeters) of the subject's eye to the display screen, as specified by the user in the ViewPoint GeometryGrid dialog.

When the head is free move and a head tracker is used, this value may not be accurate.

See also:

[VPX_GetHeadPositionAngle](#)
[VPX_GetMeasuredScreenSize](#)

```
VPX_RealPoint rsc;  
VPX_GetMeasuedViewingDistance( &rsc );
```

```
int VPX_GetMeasuredScreenSize( VPX_RealPoint *displaySize );
```

Provides the physical size of the display screen (Millimeters), as calculated from the ViewPoint GeometryGrid dialog specifications.

See also:

[VPX_GetMeasuredViewingDistance](#)

```
VPX_RealPoint rp ;  
VPX_GetMeasuredScreenSize( &rp );  
printf("Stimulus Display Screen Size is %d X %d", rp.x, rp.y );
```

15.8.7 Get ROI

```
VPX_GetROI_RealRect ( int roiN, VPX_RealRect *rr )  
  where roiN is in { 0 to 99 }
```

Retrieves the normalized floating point coordinates for the specified region of interest (ROI).

```
VPX_RealRect rr;  
RECT cr;  
INT w, h;  
GetClientRect( hwnd, &cr ); w = cr.right; h=cr.bottom;  
for ( ix=0; ix<MAX_ROI_BOXES; ix++) {  
  VPX_GetROI_RealRect( ix, &rr );  
  printf("ROI %d = (%d,%d) (%d,%d)",  
    (int)(w*rr.left), (int)(h*rr.top), (int)(w*rr.right), (int)(h*rr.bottom) );  
}
```

```
int VPX_ROI_GetHitListLength (VPX_EyeType eye )
```

The ROI may be overlapped, so a gaze point may be in more than one ROI. This function returns a count of the number of ROI the gaze point is in. This is calculated for each eye since the gaze point may be different for the left and right eyes. If the gaze point is not in any ROI, a zero value is returned. If the gaze point was in three overlapping ROI, the value three is returned, etc.

Monocular *ViewPoint* users should specify Eye_A.

See sample code under: VPX_ROI_GetHitListItem

```
int numberOfRegionsHit = VPX_ROI_GetHitListLength( EYE_A );
```

```
int VPX_ROI_GetHitListItem ( EyeType eye , int NthHit )
```

The ROI may be overlapped, so a gaze point may be in more than one ROI. This function returns the ROI index number of the Nth ROI that the gaze point is in for either Eye_A, or Eye_B if binocular mode is on; monocular *ViewPoint* should specify Eye_A. The NthHit argument starts at zero. The functions may be called repeatedly to obtain successive ROI index values until no more ROI are in the hit list. If the *NthHit* argument is greater than the hit count, then the function returns the value ROI_NOT_HIT. The test is for values inside the ROI box values, not resting on the box lines.

Monocular *ViewPoint* users should specify Eye_A.

See also: [VPX_ROI_GetHitListLength](#)

```
int ix = 0; // range is: 0 to (VPX_ROI_GetHitListLength(EYE_A) - 1 )
while( ROI_NOT_HIT != ( roiNumber = VPX_ROI_GetHitListItem( EYE_A, ix++ )))
    doSomethingWith( roiNumber );
```

```
int VPX_ROI_MakeHitListString ( EyeType eye , int NthHit )
```

The string lists the ROI that were hit, eg: "2,45,88". If (indicateOverflow == true) then a "+" at the end of the string will be used to indicate that there were additional roi that could not fit in the given string.

Makes a string with at most maxStringLength characters. Specifying a maxStringLength of 2 will effectively limit the reported roi to the first. For double digit numbers, prints both digits if possible, otherwise prints nothing. Does not leave dangling comma separators.

If there are no roi hit, then the *noHitsString* is returned, eg: "-None- "

The return value is the length of the string.

Monocular *ViewPoint* users should specify Eye_A.

See also: [VPX_ROI_GetHitListLength](#)

```
int ix = 0; // range is: 0 to (VPX_ROI_GetHitListLength(EYE_A) - 1 )
while( ROI_NOT_HIT != ( roiNumber = VPX_ROI_GetHitListItem( EYE_A, ix++ )))
    doSomethingWith( roiNumber );
```

15.8.8 Set Remote EyeImage

```
int VPX_SetEyeImageWindow( VPX_EyeType eyn, HWND hWnd );
```

Specifies the window within a layered application that should be used for display of the EyeCamera image.

See also:

[vpx_EyeCameraImageOverlays](#)
[VPX_SetEyeImageDisplayRect](#)

```
HWND hWnd = myEyeWindow ;
VPX_SetEyeImageWindow( EYE_A, hWnd );
```

```
int VPX_SetEyeImageDisplayRect( VPX_EyeType eyn, RECT displayRect );
```

Allows optional re-adjustment of the display image offset and size, from the default. NOTE:
320x240 provides optimal performance, other sizes may increase CPU usage.

See also:

[VPX_SetEyeImageWindow](#)

```
RECT displayArea = { 10, 10, 130, 250 } ;  
VPX_SetEyeImageDisplayRect( EYE_A, displayArea );
```

15.9 DLL Interface

Table 13: SDK API & Messaging Functions

VPX_InsertMessageRequest (HWND hWnd, UINT msg)

Inserts the specified hWnd into the list of windows that are sent notification messages. E.g. when fresh data has been put in the DLL shared memory. Refer to Table 40

VPX_RemoveMessageRequest (HWND hWnd)

Removes your application's request for notification for the specified window.

VPX_GetMessageListLength (int * num)

Returns the number of windows that are registered to receive messages.

VPX_GetMessagePostCount (int * num)

Returns the total number of messages that have been distributed.

int VPX_GetViewPointAppCount(int *apps);

Sets applications to non-zero if ViewPoint is running.

BOOL VPX_VersionMismatch (VPX_SDK_VERSION)

Returns 0 if the program was compiled with the same version of the DLL lib as the DLL lib that is loaded at run time.

double VPX_GetDLLVersion()

You can obtain the version number of the loaded DLL using this function.

Example:

```
double dllVersion = VPX_GetDLLVersion();
if ( VPX_SDK_VERSION != dllVersion ) doSomething();
```

VPX_STATUS_DistributorAttached

The DLL based SDK gets data from, and sends command strings to, a "distributor" application.

Normally the distributor application is the ViewPoint EyeTracker, but it could be the RemoteLink application. We now provide a mechanism for determining which, if any, it is.

```
#define VPX_Distributor_None           0
#define VPX_Distributor_IsViewPoint   1
#define VPX_Distributor_IsRemoteLink  2
#define VPX_DistributorType           int
```

```
VPX_DistributorType dllDataSource = VPX_GetStatus(
VPX_STATUS_DistributorAttached );
```

// Note: VPX_STATUS_ViewPointIsRunning returns true if ViewPoint is running either directly or via RemoteLink.

Table 14: SDK Utility Functions

VPX_GetPrecisionDeltaTime (double*, resetHoldTime)

Retrieves the delta time between the holdTime and the current time.

BOOL VPX_IsPrecisionDeltaTimeAvailableQ()

Use this command to determine if precision time is supported. Returns TRUE if the system supports precision time, otherwise returns false.

RectFrame (HDC hdc, int x1, int y1, int x2, int y2, int t)

Draws two concentric hollow rectangles in the specified window. The inner rectangle is defined by the specified coordinates. The outer rectangle is larger by parameter **t** pixels.

EllipseFrame (HDC hdc, int x1, int y1, int x2, int y2, int t)

Draws two concentric hollow ellipses in the specified window. The inner rectangle is defined by the specified coordinates. The outer rectangle is larger by parameter **t** pixels.

VPX_WindowRECT2RealRect(RECT nr, RECT clientRect, RealRect * rr);

Takes in integer coordinates for a rectangle within a specified window and returns the normalized coordinates for that rectangle.

VPX_RealRect2WindowRECT(RealRect rr, RECT clientRect, RECT * scaledRect);

Takes normalized coordinates of a rectangle and returns integer coordinates that have been scaled for the size of the specified window. For example:

```
RealRect rr = { 0.1, 0.1, 0.2, 0.2 }; RECT clientWindowRect = { 320, 240 }; RECT
scaledRect ; VPX_RealRect2WindowRECT( rr, clientWindowRect, &scaledRect );
scaledRect will now contain { 32, 24, 64, 48 }
```

VPX_drawROI (HWND hWnd, int activeRegion)

Draws the activeRegion ROI in red and all of the other ROI in blue, within the specified window.

int VPX_LParam2RectPoint (LPARAM codedLoc, RECT clientRect, POINT *pt);

Used with VPX_CAL_* messages to obtain the location of the calibration point that is encoded in the message LPARAM. Takes LPARAM and returns integer coordinates of the calibration points that have been scaled for the size of the specified window. Previously defined in DLL but not listed in prototypes, because it was under evaluation. Added here in version 2.4.2.0

15.10 ViewPoint Events & Notification Messages

15.10.1 General Events

HIWORD(WPARAM)	VPX_DAT_FRESH The data has just been updated, real-time programs should now access the data that it needs by calling the accessor functions.
LOWORD(WPARAM)	The eye the command pertains to: EYE_A, EYE_B <pre>VPX_EyeType eyn = (VPX_EyeType)LOWORD(wparam); VPX_RealRect gpt; VPX_GetGazePoint2(eyn, &gpt);</pre>
LPARAM	Do not use LPARAM.

HIWORD(WPARAM)	VPX_ROI_CHANGE Indicates that a Region Of Interest (ROI) was changed.
LOWORD(WPARAM)	<pre>RealRect rr ; RECT cr, dr ; GetClientRect(hwnd, &cr); WORD roiIndexNumber = LOWORD(wParam); VPX_GetROI_RealRect(roiIndexNumber, &rr); VPX_RealRect2WindowRECT(rr, cr, &dr); Rectangle(hdc, dr.left, dr.top, dr.right, dr.bottom);</pre>
LPARAM	Do not use LPARAM.

HIWORD(WPARAM)	VPX_STATUS_CHANGE Indicates that a key ViewPoint status item was changed. For details, see Section: 15.8.4: Get ViewPoint Status. VPX_GetStatus
LOWORD(WPARAM)	Do not use WPARAM.
LPARAM	<pre>WORD statusItem = LOWORD(lParam); WORD statusValue = HIWORD(lParam); switch (statusItem) { case VPX_STATUS_DataFileIsOpen : printf("DataFile is %s", (statusValue==1)?"Open":"Closed"); break; ... }</pre>

HIWORD(WPARAM)	VPX_VIDEO_FrameAvailable Notifies external (layered) applications that a video frame is available in ViewPoint memory. See also: .
LOWORD(WPARAM)	VPX_EyeType eye = LOWORD(wparam); // Usage: if (eye == EYE_A)
LPARAM	DWORD notUsed = lparam ; // NOTE: subject to change!

HIWORD(WPARAM)	VPX_VIDEO_SyncSignal This message is sent as soon as the video capture board detects frame-ready (30 Hz) or field-ready (60 Hz) signal. The user can now tell when the image became available for processing, before any image processing has been performed. This better reflects the true time of the eye movement, and reduces noise in the timing calculation. See also: vpx_event +videoSynch VPX_GetDataTime2
LOWORD(WPARAM)	VPX_EyeType eye = LOWORD(wparam); // Usage: if (eye == EYE_A)
LPARAM	DWORD deltaMicroSeconds = lparam; // NOTE: subject to change!

15.10.2 Calibration Events

The flow of the calibration events in the autocalibration sequence is as follows:

```

VPX_CAL_BEGIN
VPX_CAL_WARN

    // each calibration point
    VPX_CAL_SHOW

        // for radius 15 down to 0
        VPX_CAL_ZOOM

    VPX_CAL_SNAP
    VPX_CAL_HIDE

VPX_CAL_END

```

HIWORD(WPARAM)	<p>VPX_CAL_BEGIN</p> <p>Indicates that a calibration sequence is about to start.</p> <p>This is the first calibration message in the sequence. In general you would want to blank the stimulus display screen and disable other graphics drawing.</p>
LOWORD(WPARAM)	<p>The calibration point number, the actual point index number, not the random or custom sequence number.</p>
LPARAM	<p>The location of the upcoming stimulus point.</p> <p><i>// Contains the location of the upcoming stimulus point. As below, use: VPX_LParam2RectPoint(lParam, cr, &calPt);</i></p>

HIWORD(WPARAM)	<p>VPX_CAL_WARN</p> <p>Provides an opportunity to display a warning message to the subject, to make sure that they are paying attention.</p> <p>Follows VPX_CAL_BEGIN.</p> <p>The warmomg time, i.e., the delay between this event and the next event, can be specified in ViewPoint, EyeSpace window, Advanced button, WarningTime slider.</p>
LOWORD(WPARAM)	<p>The calibration point number, the actual point index number, not the random or custom sequence number.</p>
LPARAM	<p>Contains the location of the upcoming stimulus point.</p> <p><i>POINT calPt; PTCHAR str = " PAY ATTENTION " ; RECT cr ; GetClientRect(hwnd, &cr); VPX_LParam2RectPoint(lParam, cr, &calPt); TextOut(hdc, calPt.x-80,calPt.y, str, strlen(str));</i></p>

HIWORD(WPARAM)	<p>VPX_CAL_SHOW</p> <p>Indicates that the calibration stimulus point should be drawn. Follows VPX_CAL_WARN for the first stimulus point; loops back to here after VPX_CAL_HIDE for each additional stimulus point.</p>
LOWORD(WPARAM)	<p>The calibration point number, the actual point index number, not the random or custom sequence number.</p>
LPARAM	<p>Contains the location of the stimulus point. As above, use: VPX_LParam2RectPoint(lParam, cr, &calPt);</p>

HIWORD(WPARAM)	<p>VPX_CAL_ZOOM</p> <p>Indicates a radius change of the tunnel motion of the stimulus. Follows VPX_CAL_SHOW and is repeatedly sent until the radius shrinks to zero.</p>
LOWORD(WPARAM)	<p>The stimulus radius (shrinks from 15 to 2). <code>WORD zoomSize = LOWORD(wParam);</code></p>
LPARAM	<p>Contains the location of the stimulus point.</p> <pre>POINT pt ; WORD r, zoomSize = LOWORD(wParam); RECT cr ; GetClientRect(hwnd, &cr); VPX_LParam2RectPoint(lParam, cr, &pt); r = cr.right * zoomSize / 200 ; Rectangle (hdc, pt.x - r, pt.y - r, pt.x + r, pt.y + r);</pre>

HIWORD(WPARAM)	<p>VPX_CAL_SNAP</p> <p>Indicates that the calibration image of the eye is being taken</p> <p>Follows the series of VPX_CAL_ZOOM events, after zoomSize has shrunk to zero; or if the calibration mode is in snapMode, this is called itself.</p>
LOWORD(WPARAM)	<p>The calibration point number, the actual point index number, not the random or custom sequence number, is in the lower 8 bits, flags for slipCorrection mode and snapMode are in the upper 8 bits.</p> <pre>WORD loWordw = LOWORD(wParam); BOOL slipMode = (loWordw & 128) ? 1 : 0 ; // bit 8 BOOL snapMode = (loWordw & 256) ? 1 : 0 ; // bit 9 int pointNumber = LOWORD(wParam) & 127 ; // lower bits 0..7</pre>
LPARAM	<p>Contains the location of the stimulus point.</p> <p>As above, use: <code>VPX_LParam2RectPoint(lParam, cr, &calPt);</code></p>

HIWORD(WPARAM)	<p>VPX_CAL_HIDE</p> <p>Indicates completion of the current calibration point. The program should clean up any remnants of this last calibration stimulus point display.</p> <p>Follows VPX_CAL_SNAP.</p>
LOWORD(WPARAM)	<p>The calibration point number, the actual point index number, not the random or custom sequence number.</p> <pre>int pointNumber = LOWORD(wParam)</pre>
LPARAM	<p>Contains the location of the stimulus point.</p> <p>As above, use: <code>VPX_LParam2RectPoint(lParam, cr, &calPt);</code></p>

HIWORD (WPARAM)	<p>VPX_CAL_END</p> <p>Indicates that the entire calibration sequence has finished. The LOWORD indicates whether or not a slipFix was requested. A 1 indicates slipFix, zero indicates (re)calibration of a point. Follows the VPX_CAL_SNAP of the last calibration stimulus point.</p>
LOWORD (WPARAM)	<p>Indicates whether or not a slipFix was requested (rather than e.g. a recalibration) . A 1 indicates slipFix, zero indicates recalibration of a point. <i>Note:</i> this is not currently consistent with the wparam format used in VPX_CAL_SNAP, but it may be made consistent in the future.</p> <p><code>BOOL doSlipFix = LOWORD(wParam) == 1 ;</code></p>
LPARAM	<p>Contains the location of the stimulus point. <i>As above, use:</i> <code>VPX_LParam2RectPoint(lParam, cr, &calPt);</code></p>

Chapter 16 Legacy, Obsolete, & Deprecated

Do not use the following for new work. They are described here only for reference use with already existing code and to provide a migration path for new code development.

16.1.1 Old CLP

Old Name	New Name	Reason
<code>circularPupilCriteria</code>	<code>pupilAspectCriterion</code>	Singular / Clarity
<code>calibrationSpeed</code>	<code>calibration_StimulusDuration</code>	Clarity
<code>gazeGraphics</code>	<code>gazeGraphicsOptions</code>	Easier string parser
<code>stimulusGraphics</code>	<code>stimulusGraphicsOptions</code>	Easier string parser
<code>pupilMinWidthCriteria</code>	<code>pupilMinWidthCriterion</code>	Singular
<code>pupilAspectCriteria</code>	<code>pupilAspectCriterion</code>	Singular
<code>circularPupilCriteria</code>	<code>pupilAspectCriterion</code>	Clarity & Singular
<code>pupilMaxWidthCriteria</code>	<code>pupilMaxWidthCriterion</code>	Singular
<code>cursorControl</code>	<code>cursor_Control</code>	Consistency
<code>gazeColor</code>	<code>penColorA</code>	Clarity & binoc

16.1.2 Old VPX

Old

`VPX_GetGlintScanOffset`
`VPX_GetGlintScanUnyokedOffset`
`VPX_CHANGE_GlintScanOffset`
`VPX_CHANGE_PupilScanArea`
`VPX_CHANGE_GlintScanSize`
`VPX_SetCalibrationDensity`

`VPX_LaunchViewPoint`
`VPX_LaunchViewPointEx`

`VPX_DataFile_StoreRejectData`
`VPX_DAT_FAILED`

`VPX_GetROI_InCode`
`VPX_DisplayROI_InCode`

Change

Obsolete

Obsolete

Use: `VPX_LaunchApp`

In previous versions of ViewPoint, the default was to only store good data in the data file. This command was used to override that default behavior. In newer versions of ViewPoint all data is stored together with a QualityCode that indicates how good the data is. See:

[VPX_GetQualityCode](#).

Old positional bit code was cumbersome.
New string parser is easier and clearer

Chapter 17 Troubleshooting

This section discusses some of the common sources of error and problem areas. Once recognized, many of these can be avoided.

17.1 EventHistory Window

The **EventHistory** window can be a very useful tool for troubleshooting many problems including video and settings file problems. Use menu item **Windows > EventHistory** to view. For troubleshooting settings file commands menu item **File > Settings > Verbose loading** will display extra information from the CLP.

17.2 Improving Frame Rate

The video frame rate will be compromised when other demands are made on the computer. Ways to improve video frame rate include:

- Closing the **Event History** window will significantly improve performance.
- Turn off "Show Dots" in the **EyeCamera** window.
- Turn off the screen saver.
- Ensure that there are no other applications running that are not required.

17.3 EyeCamera Window Troubleshooting

If a video source was connected to the computer when *ViewPoint*TM was started, the **EyeCamera** window should display the captured video image. Otherwise, follow the following troubleshooting tips:

- If the **EyeCamera** window shows "*** FROZEN ***", then you should select the menu item: **Video > Freeze Video**, which will toggle the check mark next to this item and unfreeze the video processing.
- Ensure that the frame grabber board and drivers have been correctly installed. Check in the Windows Device Manager for conflicts.
- Reset the video: **Video > Reset Video**

Note: If a camera is disconnected and reconnected then ViewPoint will automatically reset the video pipe and start working again. If after eight reset tries the software cannot detect a camera then the message MaxVideoResetTries exceeded will appear in the Status window.

- If the **EyeCamera** window background is black, white or blue, then check the

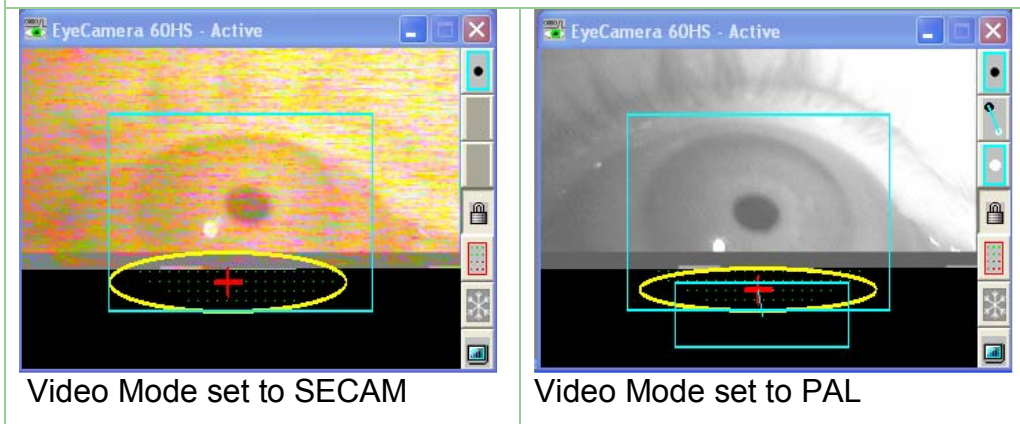
following:

- The camera is plugged into the computer properly.
- The camera is getting power that it needs, e.g., from a power supply.
- The camera is getting enough light.
- The camera iris adjustment is open.
- The lens cap has been removed!
- If the **EyeCamera** window video segmentation is not working, make sure the display monitor is set to True Color (32 bit).

17.3.1 Bottom half of EyeCamera window is black

If the bottom of **EyeCamera** image is black as in Figure 19: Then the video standard has been set to PAL or SECAM, but the camera is NTSC. All ARI supplied cameras are NTSC. Select menu item: **Video > Video Standard > NTSC**. Also check any Settings file that may be loaded, e.g. StartUp.txt, that may specify a different video standard.

Figure 19: EyeCamera Window if Incorrect Video Mode Selected



17.4 General Troubleshooting

If the color of the sliders and buttons are different than the window background, change the appearance settings in the **Windows Controls > Display > Appearance** to the Windows Standard scheme.

Chapter 18 History of Eye Tracking Methods

The quest to be able to determine where the eyes are looking has been long and elusive. Many talented individuals have invested many years to achieve this goal and many methods have been tried. It is useful to understand some of the methods available, so as to avoid repeating mistakes and to choose the best method for a particular purpose.

18.1 Electrical Methods

18.1.1 Surface Recordings

The most obvious solution suggested by most lay people is to record the eye muscle activity around the eye, but this electromyographic information is insufficient to determine the position of gaze. Interestingly however, there is an electrical potential between the front and the back of the eyeball. Measurement of this potential is called electro-oculography (EOG). It is relatively easy, but not very precise. One of the problems is the potential shows substantial diurnal variation, which necessitates that experiments be conducted at the same time of day. This method can also show substantial drift of the signal over time.

18.1.2 Induction Coils

With this method the head must be inside of a box frame that holds large magnetic induction coils, which bathe the head in alternating magnetic fields. The different dimensions of the box use different alternation frequencies. Electrical currents will be induced in a coil of wire that is moved inside box. Movements in different dimensions can be de-multiplexed by selectively filtering for the different alternation frequencies. Permanently implanting coils of wire in the eyes, so called *scleral search coils*, provides one of the most accurate methods of eye tracking available to date. Needless to say, this is usually possible only in animal experiments. Alternatively, tight fitting contact lenses can be used in humans, but the lead-wires hanging from the contact lenses interfere with normal eye blinks and they can not be tolerated for very long.

18.2 Optical Methods

18.2.1 Reflections, or Purkinje Images

Light is reflected from surfaces when there is a change in optical density. This occurs in the eye first at the corneal surface (air to cornea), second from the back of the cornea (cornea to aqueous humor), third at the front surface of the lens and fourth from the back surface of the lens. These reflections are referred to as the first to fourth Purkinje images, respectively. These reflections can be used for eye tracking.

Corneal Reflection Tracking

The first purkinje image, the reflection from the front of the cornea, is also referred to as glint. An infrared light source produces a specular reflection on the smooth cornea, which is recorded by

an infrared light sensitive device. Used alone, this method is very sensitive to head movement when calculating direction of gaze.

Other Reflections

The other Purkinje images can also be used for eye tracking and they can be used in combination with one another. One problem is that the higher numbered (deeper within the eye) Purkinje images tend to be quite dim compared to the first Purkinje image.

18.2.2 Dark Pupil Tracking

An un-collimated infrared light source will make even the darkest iris appear light, so as to produce a high contrast with the dark pupil that acts as a sink for the infrared light. The pupil edges are located and the pupil center is calculated. Used alone, this method is sensitive to head movement when calculating direction of gaze.

18.2.3 Limbus Tracker

The limbus is the junction between the smooth clear cornea and the much rougher white sclera that surrounds it. This method takes advantage of there being a difference in the amount of light reflected from the cornea compared to the sclera. This reflectivity difference produces a contrast difference that can be monitored by photodetectors (e.g. phototransistors, or historically photodiodes); typically two photodetectors are placed on either side of the eyeball. Used alone this method is sensitive to head movement when calculating direction of gaze.

18.2.4 Bright Pupil Method

Collimated infrared light reflects off the retina, similar to the reflection we see from the eyes of a nocturnal animal, or in red-eye from flash photography. This also can be detected and located. Used alone, this method is sensitive to head movement when calculating direction of gaze.

18.2.5 Corneal Bulge Method

It is possible to calculate the location of the corneal bulge by using an array of detectors placed around the eye to sense variations in total infrared reflection. This system has the advantage of being able to locate the bulge even when the eye is closed, however it is typically confused by eye blinks. Used alone, this method is sensitive to head movement when calculating direction of gaze.

18.2.6 Vector Difference Method

When using only a single signal there is always confusion between eye movements and head movements. Most eye movements are relatively small compared to head movements, even when a person thinks that they are holding their head still. A solution is to use two signals that move together in a constant way when the head moves, but that vary from one another as the eye moves. By comparing only the difference between these two signals, eye movements can be disambiguated from head movements. This difference can be thought of as a floating vector. Only the magnitude and direction are important, not the absolute position. We will briefly discuss two of these *vector difference methods*, both of which usually employ video image processing.

One popular vector difference method compares the corneal reflection, i.e., the first Purkinje image, to the reflection from the back surface of the crystalline lens, i.e., the fourth Purkinje image,

and is often referred to as a Purkinje eye tracker. The fourth Purkinje image is however quite dim and care must be taken not to expose the subject to excessive amounts of infrared light in an attempt to image it.

The most easily observed vector difference method is sometimes called the Pupil-Corneal Complex Method, which is the method used by ViewPoint™. In this method it is the difference between the position of the corneal reflection and the position of the pupil.

Vector difference methods are not without problems of their own. (a) There are now two sources of position noise instead of one. (b) Given a change in viewing direction, the magnitude of the vector signal is smaller than that of the individual signals; the result is a lower signal to noise ratio. (c) While the vector difference methods are robust against horizontal (sideways, x-axis) and vertical (up/down, y-axis) direction movements, they are more sensitive to in-and-out (closer or farther from the camera, z-axis) movement of the head. This is because the distance between the two points, i.e., the vector length, becomes shorter in video image, as the head is moved backward away from the camera.

Chapter 19 Binocular Option

This section describes those features that are particular to the *ViewPoint EyeTracker*[®] binocular option only.

19.1 Installing Binocular FrameGrabber & Software

Refer to [Chapter 3](#).

19.2 Operating in Binocular Mode

The main (monocular) ViewPoint window shows Eye-A. When in binocular mode, the second *ViewPoint* window shows Eye-B. To switch from monocular to binocular operation, select menu item: **Binocular > Binocular Mode**

Alternatively, use the command line parser (CLP) command:

```
binocular_Mode [ On , Off ]
```

This command may be placed in the settings file `startup.txt` to automatically set your binocular preference when ViewPoint is launched. Refer to **Chapter 10** and **Chapter 11**.

When in binocular mode, the Eye-B window contains only those controls and menu items necessary to adjust the [EyeCamera](#) image settings and calibration points for Eye-B. All other controls will be made through the Eye-A window.

19.3 Setup

For optimal performance in the binocular mode, follow these suggestions:

The video modes may be different between Eye-A and Eye-B during setup. However, when collecting data both Eye-A and Eye-B are collecting at the same speed.

Set the video modes to High Precision rather than High Speed, unless the higher speed is actually required.

Close [EyeCamera](#) windows in both Eye-A and Eye-B when running in High Speed mode.

Many controls are disabled in the Eye-B window. Do not try to bypass the inactivated menu items by loading settings files that contain commands to change these items.

When collecting data, avoid moving the mouse, especially to resize or move windows, which may cause video frame loss especially on slower computers.

19.4 Storing Data

The data file will automatically have data columns appended for binocular mode. Quality markers are recorded for each eye which will allow collection of data from one eye, even if the

other eye data has been rejected for some reason. The data file record format is typically a sequence as follows:

```
Tag#, EyeA_data, EyeB_data, Count, Markers
```

Refer to [Chapter 9](#) for details of data file format.

19.5 Real-Time Display of Binocular Data

The binocular data may be combined in a “cyclopean” average position during the display in the [GazeSpace](#) and [Stimulus](#) windows and in the [PenPlot](#) window:

See menu items:

Binocular > Show both eye positions

Both eye positions will be displayed as separate points in the [GazeSpace](#) and [Stimulus](#) windows.

Binocular > Show Averaged Y-Positions

Two positions of gaze will be displayed in the [GazeSpace](#) and [Stimulus](#) windows. These will reflect the average of the vertical (Y) positions of both eyes.

Binocular > Show Average of Eye Positions

One position of gaze will be displayed in the [GazeSpace](#) and [Stimulus](#) windows. This will take the average of the eye positions.

Note: Regions of Interest (ROI) hit lists are triggered by the (possibly smoothed) individual positions of gaze, but with no binocular averaging.

Alternatively, use the command line parser (CLP) command:

```
binocular_Averaging [Off, only_Y, both_XY ]
```

This command may be placed in the settings file `startup.txt` to automatically set your binocular preference when ViewPoint is launched. Refer to [Chapter 10](#) and [Chapter 11](#).

19.6 Interfacing to Other Applications

ViewPoint EyeTracker[®] provides for communication with other applications running on the same computer or a remote computer. The software developer kit (SDK) contains new or augmented functions for binocular data typically taking the following form:

```
New: VPX_SomeFunction2 ( EyeType eye, VarType *var );
```

```
Old: VPX_SomeFunction (VarType *var )
```

```
#define EyeType int
```

```
#define Eye_A 0 // the first, or monocular eye
```

```
#define Eye_B 1 // the second, or binocular option eye
```

The SDK notification message VPX_DAT_FRESH is issued separately for each eye. The



low word of the message contains the eye of origin information:

```
EyeType eye = LOWORD (wParam) ; // will be either Eye_A or Eye_B
```

Note: Serial port packets and the RemoteLink application currently do not contain information about the second eye. Updates will be sent to customers when this is included.

19.7 Settings Files

You will need to load and save separate settings files for each eye using menu items: [File > Data >](#) Refer to Chapter 10.

Chapter 20 Head Tracker Option

This section describes those features that are particular to the *ViewPoint EyeTracker*[®] head tracker option only.

20.1 General

Currently the *ViewPoint EyeTracker*[®] works exclusively with the FlockOfBirds (FOB) magnetic tracker for its head tracking capability. *ViewPoint* connects to FOB via a serial port interface. We have designed the interface so that the user should not need to spend much time reading the FOB Installation and Operation Guide (IOG). In particular, you do not need to examine or change the internal jumpers inside the FOB electronics unit, you do not need to consider the low level commands that are described, and any dip switch changes that are pertinent to *ViewPoint* users are described below. Note: if you read only the sections of the IOG specified below, and SKIP the other sections, you should stay out of trouble.

Important: The sensor must be within +/- 4 feet of the transmitter; otherwise, data will not be accurate.

20.2 Unpacking

Please check the contents of the FOB box listed in section 2.0. Items 1 through 4 and 7 through 10 should be included. Items 5 and 6 are not part of this package and the optional extended range transmitter is not included.

20.3 Installation

Please read the following sections of the IOG:

- 2.1.1 ELECTRONICS UNIT LOCATION
- 2.2.2 TRANSMITTER LOCATION
- 2.1.3 SENSOR LOCATION
- 2.1.4 POWER SUPPLY LOCATION

Please SKIP sections 2.1.5, 2.1.6, and 2.1.7, as these are preset for *ViewPoint*.

20.4 Cable Attachment

Please connect the cables as described on p.15-16, in section 2.2:

- 2.2.1 RS-232C CABLE
- 2.2.4 SENSOR CABLE

- 2.2.5 TRANSMITTER CABLE
- 2.2.6 CRT SYNC CABLE (optional)
- 2.2.7 POWER CABLE

Sections 2.2.2 and 2.2.3 relating to FBB CABLE are not relevant.

20.5 Baud Rate and DipSwitch Settings

Currently the *ViewPoint EyeTracker*[®] works only with Position/Angles data records (this may be changed in the future to allow for additional capabilities). The Position/Angles data record type allows 113 records per second at 19.2K baud, and approximately 56 records per second at 9600 baud. The current default is 19.2K. In any case, the FOB settings and the ViewPoint settings must match. The baud rate for the FOB is set by adjusting the dipswitches on the back of the FOB box. Dipswitch changes must be made when the switch on the front of the FOB box is set to STBY.

The dipswitch settings are described on p.12 of the FOB-IOG, and are summarized here.

Table 15: Head Tracker Dip Switch Settings			
DIPSWITCH #			BAUD RATE
1	2	3	
Off	On	On	9600
On	Off	Off	19.2K

All other dipswitches (4 through 8) should remain in the up (OFF) position.

The baud rate is specified via the dialog box: *ViewPoint* menu item: **HeadTrack > Connection Settings...** Only the port and the baud rate should be modified; all other settings should remain the same. DataBits=8, Parity=None, StopBits=1, and all check boxes must be left unchecked.

Note: Depending on the type of computer you are using, when the eye tracker is running at 60Hz Binocular, the frame rate may be less noisy if the baud rate to the FOB is lowered from 19.2K baud to 9600 baud.

20.6 Connection to FOB

After all port and baud rate settings have been specified, the user may connect to the FOB.

First flip the FOB front panel switch to FLY, the red light on the front of the box should will start blinking as it is trying to find the small sensor, then the red light should be constantly ON.

Next, use the *ViewPoint* menu item: **HeadTrack > Connect**. The red light on the front of

the box may or may not blink, but after a few seconds, it should be constantly ON. If it is not, there is a connection problem; check baud rate match, cables, etc.

After connecting the first time, the user may wish to select *ViewPoint* menu item: [HeadTrack > Disconnect](#) when not collecting data, because this will put the FOB to sleep and turn off the strong magnetic field.

20.7 CRT Synchronization

ViewPoint has been designed to take advantage of the FOB CRT SYNC option. If working close to a CRT, the magnetic fields can interfere with the display. You can synchronize the FOB to the monitor using the CRT SYNC Pickup. First find a location where the signal is strong; the maximum voltage will be obtained on the top or side of the CRT housing near the deflection yoke, typically halfway between the front and the back of the cabinet.

Make sure the pickup is plugged into the FOB before starting. See p. 16, section 2.2.6 of IOG for details.

Select *ViewPoint* menu item: [HeadTracker > Connect](#)

Select *ViewPoint* menu item: [HeadTracker > CRT Sync > Test Signal Strength](#)

The voltages and the CRT refresh rate will be displayed in the [Status](#) window and in the [EventHistory](#) window. The FOB light should not be glowing when you are in this mode.

Important: This voltage and Hz data should be correct when connected at 19.2K baud, but may be incorrect when connected at 9600 baud.

Select *ViewPoint* menu item: [HeadTracker > CRT Sync > Sync to 50- 72 Hz Display](#), or select [Sync to 73-144 Hz Display](#), depending on your CRT display refresh rate.

20.8 Location and Orientation of Transmitter & Receiver

The penPlot ranges assume that the transmitter is at approximately the left shoulder and (design side) facing right. The receiver should be on the head in the same orientation, with the wire leaving it to the left and the design side facing right. The bottoms of each should be downward.

In the above configuration the data file columns will be as follows:

Position Coordinates (cm):

- X-pos will indicate horizontal movement with positive to the right
- Y-pos will indicate vertical movement with positive upward
- Z-pos will indicate movement of the head toward or away from the CRT display

Orientation Angles (degrees):

- Yaw about the Y-axis. (The position where the nose points moves along the horizontal.)
- Pitch about the X-axis. (The position where the nose points moves along the vertical.)
- Roll about the Z-axis.

20.9 Head Tracker Event Data

The FOB hardware always creates new data at exactly 100Hz; this cannot be modified. We can request that data be sent to the computer at various rates, including rates that are faster than new data is being created, but this would be a waste of computer resources to process the incoming data. The timing of the eye tracker cameras is not synchronized with the timing of the head tracker, so fresh data is arriving from these various sources at different times.

Select the menu item: [HeadTracker > Asynchronous Data Storage](#) if you want the incoming FOB data to be inserted into the data file as soon as it is acquired, which will have it appear on a separate data line. Otherwise, uncheck this option to have the most current head tracker data appended to the each line of eye tracker data (i.e., in the same record). The asynchronous head tracker data is indicated by record tag number 14 in the first column of the data file.

Refer to [Chapter 9](#) for data file format.

20.10 HeadSpace Window

The [HeadSpace](#) window provides real-time graphical display of head position and head angular orientation. The window also displays the stimulus display screen (x,y) intercept point of the (gaze forward) primary axis of the eye, based on the user specified distance from the headTracker sensor to the center of rotation of eye.

20.11 Troubleshooting

If there is no movement showing in the data, briefly switch to STBY and then back to FLY



ARI Software License

PLEASE READ THIS SOFTWARE LICENSE AGREEMENT "AGREEMENT" CAREFULLY BEFORE USING THE SOFTWARE. BY USING ALL OR ANY PART OF THE SOFTWARE, YOU ARE AGREEING TO BE BOUND BY ALL OF THE TERMS AND CONDITIONS OF THIS AGREEMENT. IF YOU ACQUIRED THE SOFTWARE ON TANGIBLE MEDIA (e.g. CD) WITHOUT AN OPPORTUNITY TO REVIEW THIS AGREEMENT AND YOU DO NOT ACCEPT THIS AGREEMENT, YOU MAY OBTAIN A REFUND OF ANY AMOUNT YOU ORIGINALLY PAID IF YOU: (A) DO NOT USE THE SOFTWARE AND (B) RETURN IT, WITH PROOF OF PAYMENT, TO THE LOCATION FROM WHICH IT WAS OBTAINED WITHIN THIRTY (30) DAYS OF THE PURCHASE DATE.

1. Definitions: "Software" means (a) all of the contents of the files, disk(s), CD-ROM(s) or other media with which this Agreement is provided, including but not limited to (i) ARI or third party computer information or software; (ii) digital images, stock photographs, clip art, sounds or other artistic works ("Stock Files"); and (iii) related explanatory written materials or files ("Documentation"); and (b) upgrades, modified versions, updates, additions, and copies of the Software, if any, licensed to you by ARI (collectively, "Updates"). "Use" or "Using" means to access, install, download, copy or otherwise benefit from using the functionality of the Software in accordance with the Documentation. "Permitted Number" means one (1) unless otherwise indicated under a valid license (e.g. volume license) granted by ARI. "Computer" means an electronic device that accepts information in digital or similar form and manipulates it for a specific result based on a sequence of instructions. "ARI" means Arrington Research, Inc., an Arizona corporation.

2. Software License: As long as you comply with the terms of this Agreement, ARI grants to you a non-exclusive license to use the Software for the purposes described in the Documentation. You may install and use a copy of the Software on your compatible computer, up to the Permitted Number of computers; You may make one backup copy of the Software, provided your backup copy is not installed or used on any computer. The software accompanying this Agreement, whether on disk, on compact disk, in read only memory, or any other media, the related documentation and other materials (collectively, the "ARI Software") are licensed, not sold, to you by ARI. The ARI Software in this package and any copies, modifications and distributions which this Agreement authorizes you to make are subject to this Agreement.

3. Intellectual Property Rights. The Software and any copies that you are authorized by ARI to make are the intellectual property of and are owned by ARI. The structure, organization and code of the Software are the valuable trade secrets and confidential information of ARI. The Software is protected by copyright, including without limitation by United States Copyright Law, international treaty provisions and applicable laws in the country in which it is being used. You may not copy the Software, except as set forth in Section 2 ("Software License"). Any copies that you are permitted to make pursuant to this Agreement must contain the same copyright and other proprietary notices that appear on or in the Software. You also agree not to reverse engineer, decompile, disassemble or otherwise attempt to discover the source code of the Software. Trademarks can only be used to identify printed output produced by the Software and such use of any trademark does not give you any rights of ownership in that trademark. Except as expressly stated above, this Agreement does not grant you any intellectual property rights in the Software. The ARI software may only be used by you for the purpose described herein and may not be disclosed to any third party or used to create any software which is substantially similar to the expression of the Software.

4. Transfer. You may not rent, lease, sublicense or authorize all or any portion of the Software to be copied onto another user's computer except as may be expressly permitted herein. You may, however, transfer all your rights to use the Software to another person or legal entity provided that: (a) you also transfer each this Agreement, the Software and all other software or hardware bundled or pre-installed with the Software, including all copies, Updates and prior versions, to such person or entity; (b) you retain no copies, including backups and copies stored on a computer; and (c) the receiving party accepts the terms and conditions of this Agreement and any other terms and conditions upon which you legally purchased a license to the Software; (d) you obtain prior written permission from ARI. Notwithstanding the foregoing, you may not transfer education, pre-release, or not for resale copies of the Software.

5. Limited Warranty on Media: ARI warrants to the person or entity that first purchases a license for the Software for use pursuant to the terms of this agreement, that the software is recorded to be free from defects in materials and workmanship under normal use for a period of ninety (90) days from the date of original purchase. Non-substantial variations of performance from the Documentation does not establish a warranty right. THIS LIMITED WARRANTY DOES NOT APPLY TO UPDATES, OR NOT FOR RESALE (NFR) COPIES OF SOFTWARE. To make a warranty claim, you must request a return merchandise authorization number, and return the Software to the location where you obtained it along with proof of



purchase within such ninety (90) day period. The entire liability of ARI and your exclusive remedy shall be limited to either, at ARI's option, the replacement of the Software or the refund of the license fee you paid for the Software. THE LIMITED WARRANTY SET FORTH IN THIS SECTION GIVES YOU SPECIFIC LEGAL RIGHTS. YOU MAY HAVE ADDITIONAL RIGHTS WHICH VARY FROM JURISDICTION TO JURISDICTION.

6. Disclaimer of Warranty. Some of the ARI Software may be designed as alpha, beta, development, continuing development, pre-release, untested, not fully tested or research only versions of the ARI Software. Such ARI Software may contain errors that could cause failure of loss of data, and may be incomplete or contain inaccuracies. You expressly acknowledge and agree that use of the ARI Software is at you sole risk. The ARI Software is provided "AS IS" and without warranty of any kind and ARI and ARI's licensor(s) EXPRESSLY DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. ARI DOES NOT WARRANT THAT THE FUNCTIONS CONTAINED IN THE ARI SOFTWARE WILL MEET YOUR REQUIREMENTS, OR THAT THE OPERATION OF THE ARI SOFTWARE WILL BE UNINTERRUPTED OR ERROR-FREE, OR THAT DEFECTS IN THE ARI SOFTWARE WILL BE CORRECTED. FURTHERMORE, ARI DOES NOT WARRANT OR MAKE ANY REPRESENTATIONS REGARDING THE USE OR THE RESULTS OF THE USE OF THE ARI SOFTWARE OR IN TERMS OF THEIR CORRECTNESS, ACCURACY, RELIABILITY, OR OTHERWISE. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY ARI OR AN ARI AUTHORIZED REPRESENTATIVE SHALL CREATE A WARRANTY OR IN ANYWAY INCREASE THE SCOPE OF THIS WARRANTY. SHOULD THE ARI SOFTWARE PROVE DEFECTIVE, YOU (AND NOT ARI OR AN ARI AUTHORIZED REPRESENTATIVE) ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION. THE LICENSE FEES FOR THE ARI SOFTWARE REFLECT THIS ALLOCATION OF RISK. SOME JURISDICTIONS DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU.

7. Limitation of Liability. UNDER NO CIRCUMSTANCES INCLUDING NEGLIGENCE, SHALL ARI BE LIABLE FOR ANY INCIDENT, SPECIAL OR CONSEQUENTIAL DAMAGES THAT RESULT FROM THE USE OR INABILITY TO USE THE ARI SOFTWARE, EVEN IF ARI OR AN ARI AUTHORIZED REPRESENTATIVE HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME JURISDICTIONS DO NOT ALLOW LIMITATIONS OR EXCLUSION OF LIMITED LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATION OR EXCLUSION MAY NOT APPLY TO YOU. In no event shall ARI's total liability to you for all damages, losses and causes of action (whether in contract, tort (including negligence) or otherwise) exceed the license fee that you paid for the Software.

8. High Risk Activities: Effort has been made to provide a bug-free product. Nevertheless, this software is not fault tolerant and is not designed, manufactured or intended for use or resale in the operation of nuclear facilities, aircraft navigation or communications systems, or air traffic control, or medical treatment and diagnosis, or for any other use where the failure of the Software could lead to death, personal injury, damage to property or severe environmental damage ("High Risk Activities"). ARI specifically disclaim any express or implied warranty of fitness for High Risk Activities.

99 2-Jun-03 © Arrington Research, Inc. All rights reserved 100 **9. Export Law Assurances.** You agree that the ARI Software will not be exported outside the United States except as authorized by United States law. You also agree that ARI Software

that has been rightfully obtained outside the United States shall not be re-exported except as authorized by the laws of the United States and of the jurisdiction in which the ARI Software was obtained.

10. Controlling Law and Severability. This Agreement shall be governed by the laws of the United States. If for any reason a court of competent jurisdiction finds any provision, or portion thereof, to be unenforceable, the remainder of this Agreement shall continue in full force and effect.

11. Complete Agreement. This Agreement constitutes the entire agreement between the parties with respect to the use of the ARI Software and supersedes all prior or contemporaneous understandings regarding such subject matter. No amendment to or modification of this Agreement will be binding unless in writing and signed by ARI.



Third Party Licenses

From time to time, some portions of the *ViewPoint EyeTracker*® code may utilize third party libraries, or modifications thereof, that have their own License Agreements. These are included here below.

Intel License Agreement For Open Source Computer Vision Library

Copyright (C) 2000-2005, Intel Corporation, all rights reserved.

Third party copyrights are property of their respective owners.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- * Redistribution's of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- * Redistribution's in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- * The name of Intel Corporation may not be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided by the copyright holders and contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall the Intel Corporation or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

AT&T License Agreement for 2D Convex Hull code

Some 2D convex hull code is modified from what was written by Ken Clarkson. Copyright (c) 1996 by AT&T.. Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which is or includes a copy or modification of this software and in all copies of the supporting documentation for such software.

* THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. IN PARTICULAR, NEITHER THE AUTHORS NOR AT&T MAKE ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.