

ViewPoint EyeTracker®

Software User Guide



Arrington Research



Contact Information

July 12, 2010 13:55:00

2010 © Arrington Research, Inc All rights reserved

Arrington Research, Inc.
27237 N 71st Place, Scottsdale, AZ 85266
United States of America

Phone +1-480.985.5810

www.ArringtonResearch.com

ViewPoint_info@ArringtonResearch.com

ViewPoint_sales@ArringtonResearch.com

ViewPoint_support@ArringtonResearch.com

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.

ViewPoint Client™ is a trademark of Arrington Research, Inc.



CHAPTER 1. INTRODUCTION 1

1.1 Congratulations..... 1
1.2 Custom Software & Hardware Development 1
1.3 User Feedback..... 1
1.4 License Information and Conditions of Use..... 1
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 ViewPoint 3

CHAPTER 2. OVERVIEW OF VIEWPOINT 4

2.1 General Description 4
2.2 Infrared Light..... 7
2.3 Mapping to Gaze Point 7
2.4 Interfaces 8
 2.4.1 Graphical User Interface (GUI) 8
 2.4.2 Command Line Interface (CLI) 8
 2.4.3 Example of a command line: 9
 2.4.4 Important CLI Changes from Previous Versions 9
 2.4.5 Arguments 10
 2.4.6 Asynchronous Operations 10
 2.4.7 Error detection and reporting 10
 2.4.8 Parameters and Arguments..... 10
 2.4.9 Known Problems and Issues 10
 2.4.10 Software Developer’s Kit (SDK) 11
 2.4.11 ViewPoint Client™ 11
2.5 Interfaces to Third Party Products – Integrity Suite™ 12
2.6 Others 12
2.7 Sample Interface Applications 12
2.8 About Writing Layered Applications 13

CHAPTER 3. QUICKSTART GUIDE 14

CHAPTER 4. INSTALLATION AND SETUP PC-60 1

4.1 Computer System Requirements..... 1
4.2 Video Requirements 1
4.3 Using with Third Party Video Input Equipment 1



4.4	PCI Video Capture Card and Driver Installation.....	2
4.5	ViewPoint EyeTracker® Software Installation.....	4
4.6	ViewPoint License (.VPL) File	4
4.7	User Windows.....	5
4.8	Menu Navigation	6
4.9	Criteria	8

CHAPTER 5. QUICK START SECTION 9

5.1	Camera Positioning	9
5.1.1	QuickClamp Lipstick Style Camera Positioning.....	9
5.1.2	EyeFrame Camera Positioning.....	10
5.1.3	Remote System Camera Positioning.....	10
5.2	Locating the pupil – all systems	10
5.2.1	Corrective Lenses (Eye Glasses)	11
5.3	Thresholding – all systems	11
5.4	Calibration.....	12
5.4.1	Calibration (Head Fixed and HMD Systems).....	12
5.4.2	Calibration (scene camera).....	13
5.5	Stimuli	14
5.5.1	Choosing the Type of Stimulus	14
5.5.2	ViewPoint Stimulus Window	14
5.5.3	Interactive Computer Display	15
5.5.4	Scene Camera systems	15
5.6	Data Collection – All Systems.....	15
5.7	Recording Scene and Screen Movies	16
5.7.1	Recorded Movie Format.....	16
5.7.2	Display of Movie Data.....	16
5.7.3	Compression	17
5.8	Data Analysis.....	17
5.8.1	Head fixed and HMD systems.....	17
5.8.2	Playing Scene and Screen Movies.....	17
5.9	Frequently Used Settings.....	18
5.10	Preferred Window Layout	18
5.11	Accelerator Keys	18
5.12	Printing	18

CHAPTER 6. LOCATING THE PUPIL AND GLINT (ALL SYSTEMS) 19

6.1	EyeCamera Window.....	20
6.2	Feature Method	20
6.3	Simulation of Gaze	21
6.3.1	Manual Simulation.....	21



6.3.2	Pattern Simulation (only on special versions)	21
6.4	Thresholding	21
6.5	Setting the Scan Density	21
6.6	Manual Thresholding of the Dark Pupil	23
6.7	Step-by-step guide for <i>Glint-Pupil Vector</i> method	24
6.8	Noise	25
6.9	Automatic Slip Compensation	26
6.10	Feature Criteria	26
6.10.1	Pupil Aspect Criterion	26
6.10.2	Width Criteria	26
6.11	Alternative Segmentation Methods	26
6.11.1	Ellipse	26
6.11.2	Centroid	27
6.11.3	Oval Fit	27
6.11.4	Edge Trace (only on special versions)	27
6.11.5	Glint Segmentation Methods	27
6.11.6	Pupil Scan Area Shape Options	27
<u>CHAPTER 7. CALIBRATION</u>		<u>28</u>
7.1	Calibration Carryover	28
7.2	Choosing the Number of Calibration Points	28
7.3	Automatic Calibration (Head Fixed)	28
7.4	Assessing Calibration Success	29
7.5	Omitting Individual Calibration Points	30
7.6	Re-presenting Individual Calibration Data Points	31
7.7	Slip Correction	31
7.8	Parallax Correction for Binocular Scene Camera systems	32
7.8.1	Manual Adjustment	32
7.8.2	With Data	33
7.9	Dominant Eye	34
7.10	Saving Calibration Eye Images	34
7.11	Advanced Calibration Controls	34
7.11.1	Presentation Order	34
7.11.2	Snap and Increment Calibration Modes	35
7.11.3	Adjusting the Calibration Area	35
7.12	Custom Calibration Point Positions	35
7.13	Geometry Grid	37
<u>CHAPTER 8. CURSOR CONTROL</u>		<u>39</u>
<u>CHAPTER 9. OCULAR TORSION</u>		<u>40</u>



9.1	Introduction to Torsion.....	40
9.2	Procedure for Measuring Torsion.....	41
9.3	Torsion Demonstration Test	42
9.4	Overriding the Default Torsion Parameters.....	43
<u>CHAPTER 10. STIMULUS PRESENTATION (HEAD FIXED).....</u>		44
10.1	General	44
10.2	Picture Lists.....	44
10.3	Using the Stimulus Window (Head Fixed Option)	45
10.4	Using the GazeSpace Window	46
10.5	Regions of Interest (ROI).....	47
10.6	Associating an Image with Specific ROI	48
10.7	Data Smoothing	49
10.8	ROI Transition Statistics or Linkages.....	49
10.9	Using the SDK, settings files and Serial Port Interface for Stimulus Presentation	51
10.10	Integrating with Third Party Products	52
<u>CHAPTER 11. DATA COLLECTION.....</u>		53
11.1	Sampling Rate – Frame Grabber Modes PC-60	53
11.2	Saving Data to File	53
11.3	Data File Format	54
11.3.1	File header information	54
11.3.2	File records	54
11.3.3	Synchronous vs. Asynchronous data inserts	54
11.3.4	Data Record Tags.	55
11.4	Direction-of-gaze Coordinates.....	58
11.5	Raw Data.....	59
11.6	Timing Measurement	59
11.7	Region of Interest (ROI)	59
11.8	Quality Marker Codes	60
11.9	Pupil Diameter	60
11.10	Pupil Aspect	61
11.11	Display Screen Geometry	61
<u>CHAPTER 12. DATA ANALYSIS</u>		62
12.1	Real-Time	62
12.1.1	Data Smoothing	63
12.2	Fixation, Saccade, Drift and Blinks.....	63



12.2.1	Velocity Threshold	63
12.2.2	Fixations	64
12.2.3	Drift	65
12.2.4	Blinks	65
12.2.5	Events	65
12.2.6	SDK	65
12.3	Post-Hoc	65
12.4	Summary Data	66

CHAPTER 13. USING SETTINGS FILES..... 67

13.1	CLI String Parsing	67
13.2	Saving and Loading Settings Files	67
13.3	Pre-load Settings in a Startup file	68
13.4	Settings/LastRun.txt	68
13.5	Settings File Lists	68
13.6	SettingsFile Examples	68
13.7	CLI s	69
13.8	Associating CLI s with FKeys	69
13.9	Simple Command Line Interface Program	69

CHAPTER 14. ETHERNET COMMUNICATION BETWEEN PCS 70

14.1	How to use ViewPointClient	71
14.2	Third Party Applications	72
14.3	Layered Applications	72
14.4	Changing the Port Number	72
14.5	Running the Server	72
14.6	Loopback	73
14.7	Firewalls	73
14.8	Hub, Switch, Router, or a Crossover cable ?	73

CHAPTER 15. ETHERNET COMMUNICATION: PC- MAC..... 74

15.1	Overview:	74
15.2	How to use ViewPointClient for MAC:	75
15.2.1	METHOD 1	75
15.3	Third Party Applications	77
15.4	Layered Applications	77

CHAPTER 16. SERIAL PORT COMMUNICATION 78



16.1	Getting Started	78
16.2	Sending Real and Test Data	78
16.3	Transfer to Intel and Macintosh Machines.....	79
16.4	Connections	79
16.5	Serial Protocol.....	79
16.6	Serial Packet Header Structure.....	79
16.7	Serial Packet Data Structures	81
16.8	Data Value Encoding.....	82
16.9	Packet Data Structures	83
16.10	Example Serial Port Code.....	83
16.11	RemoteLink™.....	84
16.11.1	RemoteLink Setup.....	85
16.11.2	Data Transfer Formats.....	85
16.11.3	Increasing the Priority of RemoteLink	86

CHAPTER 17. VIEWPOINT INTERFACE: GUI, SDK, CLI..... 87

17.1	General	87
17.1.1	VPX_SendCommand & formatted strings	87
17.1.2	Quoting strings with white spaces.....	87
17.1.3	Case insensitive CLI strings	88
17.1.4	Boolean Toggle	88
17.1.5	SDK return values	88
17.2	Data Files	89
17.2.1	Open Data File with Randomly Generated Name	89
17.2.2	Specify NewUnique Data File Extension.....	89
17.2.3	Open a Data File and Specify a File Name	90
17.2.4	Insert a String into the Data File	90
17.2.5	Insert a Marker into the Data File.....	91
17.2.6	Insert a User Defined Data Tag into the Data File	92
17.2.7	Specifies asynchronous or synchronous string data	92
17.2.8	Specify asynchronous or synchronous marker data.....	93
17.2.9	Specify asynchronous or synchronous head tracker data	93
17.2.10	Specify data file start time.....	94
17.2.11	Store smoothed or unsmoothed data	95
17.2.12	Include Raw Eye Data	95
17.2.13	Specify whether to use buffering (DEPRECATED).....	96
17.2.14	Pause writing of data to file.....	96
17.2.15	Close Data File	97
17.2.16	Close Data File and Open in Post-Hoc Analysis tool.....	97
17.3	Stimulus Images.....	97
17.3.1	Load Stimulus Image into the Stimulus window	97
17.3.2	Specifies how to display the currently loaded stimulus image	98
17.3.3	Specify a background “matting” color for the stimulus window.....	98



17.3.4	Stereoscopic Display (Side-by-side)	99
17.3.5	Play specified Sound file	99
17.4	PictureList	99
17.4.1	Initialize Picture List	99
17.4.2	Add List of Image Names to PictureList	100
17.4.3	Randomize List of Images in the PictureList	100
17.4.4	Move to Next Image in the PictureList	100
17.4.5	Move to Start of Images in Picture List	100
17.5	Controls window: EyeImage	101
17.5.1	Specify Mapping Feature	101
17.5.2	AutoThreshold	101
17.5.3	Positive Lock Tracking	102
17.5.4	Adjust Pupil Threshold Slider	102
17.5.5	Adjust Glint Threshold Slider	103
17.5.6	Adjust Video Image Brightness	103
17.5.7	Adjust Video Image Contrast	104
17.5.8	Dynamically Optimize Brightness and Contrast Settings	104
17.5.9	Adjust Pupil Scan Density	105
17.5.10	Override Pupil Scan Density Minimum	106
17.5.11	Adjust Glint Scan Density	107
17.5.12	Override Glint Scan Density Minimum	108
17.6	EyeCamera Window	108
17.6.1	Adjust Pupil Scan Area	108
17.6.2	Specify Pupil Scan Area Shape	109
17.6.3	Pupil and Glint oval fit constraints	109
17.6.4	Define Glint Scan Area	110
17.6.5	Define Offset of Glint Scan Area Relative to the Pupil	110
17.6.6	Unyoke Glint Scan Area from the Pupil	111
17.6.7	Define offset of Unyoked Glint Scan Area	111
17.6.8	Toggle Show Treshold Dots On / Off	112
17.6.9	Specify EyeImage Overlay Graphics sent to layered application	
(EXPERIMENTAL)	112
17.6.10	EyeCamera Tool Bar Display	113
17.7	Slip Compensation	113
17.7.1	Slip Compensation Mapping Mode	113
17.7.2	Slip Compensation Speed	114
17.7.3	Slip Compensation X-Gain	114
17.7.4	Slip Compensation Y-Gain	115
17.8	Video related controls	116
17.8.1	Specify EyeCamera Video Input Standard	116
17.8.2	Specify SceneCamera Video Input Standard	116
17.8.3	Specify Tracking Operation Mode	117
17.8.4	Specify Dark or Bright Pupil Tracking	117
17.8.5	Specify Pupil Segmentation Method	117



17.8.6	Specify Glint Segmentation Method.....	118
17.8.7	Changes default setting for Freeze Feature	119
17.8.8	Toggle Freeze Video Image Preview On / Off.....	119
17.8.9	Reset Video Capture Device	120
17.9	Calibration controls	121
17.9.1	Start Auto-Calibration.....	121
17.9.2	Stop Auto-Calibration	121
17.9.3	Specify Calibration Stimulus Presentation Speed.....	121
17.9.4	Specify the duration of presentation of calibration warning notice	122
17.9.5	Specifies Interval Between Presentation of Calibration Stimulus Points	122
17.9.6	Calibration Snap Mode	123
17.9.7	RePresent in Snap Calibration Mode.....	123
17.9.8	AutoIncrement Calibration Mode	124
17.9.9	Calibration Stimulus Point Presentation Order	124
17.9.10	Specify Number of Calibration Stimulus Points.....	124
17.9.11	Specify Calibration Stimulus Point Color	125
17.9.12	Specify Calibration Stimulus Window Background Color	125
17.9.13	Randomize Calibration Stimulus Points Check Box (DEPRECATED)	125
17.9.14	Specify Calibration Stimulus Point Presentation Order.....	126
17.9.15	Specify Individual Custom Calibration Stimulus Points	126
17.9.16	Display Custom Calibration Stimulus Point Order	127
17.9.17	Select the Specified Calibration Data Point.....	127
17.9.18	Select the Index Number that Maps to the Specified Calibration Data Point 128	
17.9.19	Undo the last operation on a Calibration Data Point	128
17.9.20	Re-Present the Specified Calibration Data Point.....	129
17.9.21	Specify Custom Calibration Stimulus Point Locations	129
17.9.22	Turn Custom Calibration Stimulus Point Location ON / OFF	130
17.9.23	Print Locations of custom calibration stimulus points in EventHistory window 130	
17.9.24	Controls display of nearest-neighbor gridlines in the EyeSpace window .	130
17.9.25	Compensate for Slip	131
17.9.26	Adjust Calibration Area.....	131
17.9.27	Save Image of Eye at each Calibration Data Point.....	131
17.9.28	Specify Amount of Parallax Correction.....	132
17.9.29	Set custom calibration stimulus point based on gaze (scene) video content 133	
17.10	Controls: Criteria Controls	134
17.10.1	Specify amount of Smoothing	134
17.10.2	Specify Smoothing Algorithm to Apply.....	134
17.10.3	Specify Velocity Threshold.....	135
17.10.4	Specify amount of Drift Allowed	135
17.10.5	Specify Pupil Aspect Ratio Failure Criterion	136
17.10.6	Specify Pupil Width Failure Criterion.....	136



17.11	Region of Interest (ROI)	137
17.11.1	Define an ROI Box.....	137
17.11.2	Specify Number of ROI to be drawn in a circle around center of window 137	
17.11.3	Remove all ROI Boxes	138
17.11.4	Select a Specific ROI.....	138
17.11.5	Select the next ROI Box	138
17.11.6	Associate ROI with a particular stimulus image	139
17.11.7	Lock ROI Settings	139
17.12	PenPlot controls.....	140
17.12.1	Specify Which PenPlot Traces to Display.....	140
17.12.2	PenPlot BackGround Color	141
17.12.3	PenPlot Limen Back Ground Color.....	141
17.12.4	Specify Speed of PenPlot Scrolling	141
17.12.5	Specify Size of PenPlot Lines.....	142
17.12.6	Specify Range of PenPlot Values	142
17.12.7	Specify the behavior of the penpot after a video freeze	142
17.13	Graphics controls.....	143
17.13.1	Specify the color of the GazeSpace and PenPlot Lines.....	143
17.13.2	Specify which Overlay Graphics to Display in the GazeSpace Window...	143
17.13.3	Specify which Overlay Graphics to Display in the Stimulus Window	144
17.13.4	Erase Data Displays in the GazeSpace and Stimulus windows	144
17.13.5	Automatically erase display windows	145
17.13.6	Specify time delay for auto erase	145
17.14	Stimulus Window controls.....	146
17.14.1	Specify Stimulus Source.....	146
17.14.2	Specify Custom Stimulus window Size and Position	147
17.14.3	Automatically Show the Stimulus Window on Primary Monitor	148
17.14.4	Specify How and where to show Stimulus window.....	149
17.14.5	Calibrate to a third party application window	150
17.15	Window related controls	150
17.15.1	Print ViewPoint windows	150
17.15.2	Include Date and Time Stamp on Printed windows	151
17.15.3	Size Window	151
17.15.4	Move Window, or Move & Resize Window	152
17.15.5	Specify ViewPoint Window State, or Windows Layout	153
17.15.6	History options	154
17.15.7	Clear Event History window	154
17.15.8	Save window layout settings	154
17.16	Settings File commands	155
17.16.1	Load Settings File.....	155
17.16.2	Edit Settings File	155
17.16.3	Show Verbose Settings File Loading Details in Event History	155
17.16.4	Save Settings e.g. calibrations etc.	156



17.17	SettingsFileList commands	156
17.17.1	Initialize Settings File List.....	156
17.17.2	Next Settings File in List.....	156
17.17.3	Add Settings File to the List	156
17.17.4	Restart Settings File List.....	157
17.17.5	Toggle Autosequencer ON / OFF	157
17.17.6	Specify delay between Settings Files in List	157
17.17.7	Randomize Settings Files	158
17.18	Torsion commands	158
17.18.1	Start / Stop Torsion Calculations	158
17.18.2	Adjust Start Point of Torsion Sampling Arc	159
17.18.3	Adjust Radius of Torsion Sampling Arc.....	159
17.18.4	Autoset Torsion Template after Adjustments	160
17.18.5	Display Real-Time Torsion Data.....	160
17.18.6	Adjust Torsion Measurement Range	161
17.18.7	Adjust Torsion Measurement Resolution.....	161
17.18.8	Set Autocorrelation Template	162
17.19	Interface settings commands	162
17.19.1	Turn Cursor Control On / Off	162
17.19.2	Use Fixation to Issue Button Click	162
17.19.3	Specify Fixation Time to Issue Button Click.....	163
17.19.4	Use Blinks to Issue Button Click.....	163
17.20	RemoteLink & SerialPort controls	163
17.20.1	Connect / Disconnect Serial Port.....	163
17.20.2	Specify Serial Data to Send	164
17.20.3	Send Serial Port Ping.....	164
17.21	HeadTracking commands	164
17.21.1	Connect / Disconnect Head Tracker Interface	164
17.21.2	Specify whether to use local or global origin	165
17.21.3	Reset Origin for the Head Sensor	165
17.21.4	Set Position and Angle Origins.....	165
17.21.5	Specify the Vector between Head Sensor and the Eyeball.	166
17.21.6	Turn CRT pulse synchronization On / Off	166
17.22	Binocular commands	166
17.22.1	Turn Binocular Mode On / Off.....	166
17.22.2	Specifies Binocular Averaging.....	167
17.22.3	Specifies which eye to calibrate	167
17.23	File Related	168
17.23.1	Launch ViewPoint with Command Line Options	168
17.23.2	Quit ViewPoint.....	168
17.23.3	Specify Default Folder paths.....	169
17.24	FKey	170
17.24.1	Associate CLI s with FKeys	170
17.25	TTL.....	170



17.25.1	Associate CLI s with TTL Voltage Changes	170
17.25.2	Set TTL Output Voltages	171
17.25.3	Simulate Change in TTL Input	171
17.25.4	Print TTL values in the History Window	171
17.25.5	Set TTL Output to Indicate Data Quality Codes	172
17.26	Misc.....	173
17.26.1	Specify Verbose Information to Send to History Window	173
17.26.2	Update Eye Data on Request.....	174
17.26.3	Set Status Window Update Rate for FPS Field	174
17.26.4	SDK Debug Mode.....	174
17.26.5	Specify ViewPoint Generated Events	175
17.26.6	Turn Accelerator Key Functionality On / Off	175
17.26.7	Confirm Quit	176
17.27	Parser Instructions.....	176
17.27.1	Settings File Comment.....	176
17.27.2	End of Settings File Command.....	176

CHAPTER 18. SOFTWARE DEVELOPERS KIT (SDK)..... 177

18.1	General	177
18.2	Registering to Receive Notifications.....	177
18.3	Example SDK Code	178
18.4	Data Quality Codes	179
18.5	Sending CLI s with the SDK	179
18.6	VPX_SendCommand(“setSomething”) replaces VPX_SetSomething.....	180
18.7	High Precision Timing	180
18.8	DLL Version Checking.....	180
18.9	SDK Trouble Shooting	180
18.10	SDK Access Functions.....	180
18.10.1	Get Eye Data Access	181
18.10.2	Get Time Information	188
18.10.3	Get Motor Data	189
18.10.4	Get ViewPoint Status.....	190
18.10.5	Get ViewPoint Stimulus Window	191
18.10.6	Get ROI.....	193
18.10.7	Set Remote EyeImage.....	195
18.11	DLL Interface	197
18.11.1	General Events.....	199
18.11.2	Calibration Events.....	200
18.12	Legacy, Obsolete, & Deprecated	204
18.12.1	Old CLI.....	204
18.12.2	Old VPX.....	205



<u>CHAPTER 19.</u>	<u>TROUBLESHOOTING.....</u>	<u>206</u>
19.1	EventHistory Window	206
19.2	Improving Frame Rate	206
19.3	EyeCameraWindow Troubleshooting.....	206
19.3.1	Bottom half of EyeCamera window is black.....	207
19.4	General Troubleshooting.....	207
<u>CHAPTER 20.</u>	<u>ERROR CODES.....</u>	<u>208</u>
20.1	Introduction to Error Codes.....	208
<u>CHAPTER 21.</u>	<u>HISTORY OF EYE TRACKING METHODS.....</u>	<u>211</u>
21.1	Electrical Methods	211
21.1.1	Surface Recordings	211
21.1.2	Induction Coils	211
21.2	Optical Methods	211
21.2.1	Reflections, or Purkinje Images	211
21.2.2	Dark Pupil Tracking	212
21.2.3	Limbus Tracker.....	212
21.2.4	Bright Pupil Method.....	212
21.2.5	Corneal Bulge Method.....	212
21.2.6	Vector Difference Method.....	212
<u>CHAPTER 22.</u>	<u>BINOCULAR OPTION.....</u>	<u>214</u>
22.1	Installing Binocular FrameGrabber & Software.....	214
22.2	Operating in Binocular Mode	214
22.3	Setup.....	215
22.4	Storing Data	215
22.5	Real-Time Display of Binocular Data	215
22.6	Interfacing to Other Applications	216
<u>CHAPTER 23.</u>	<u>ARI SOFTWARE LICENSE</u>	<u>217</u>
<u>CHAPTER 24.</u>	<u>THIRD PARTY LICENSES</u>	<u>219</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 was our intention to keep both versions essentially identical. However, disproportionate customer demand for the PC version and 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: ViewPoint_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: ViewPoint_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 217. The contents of this user guide and any other documentation provided with the *ViewPoint*



Arrington Research

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

Chapter 5 – Quick Start Section A brief tutorial designed to help new users start recording eye movements very quickly and easily.

Chapter 17 is the most comprehensive reference section for all aspects of the eye tracker and should be referred to often. It contains every feature and control available, including those not described elsewhere.

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
stimulusWindowDimensions	Program variable, CLI or SDK code
Load PICT image	GUI controls: Menu Items, Buttons, Sliders.
Video window	Program Window
Figure 2	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: ViewPoint_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).

Please also 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 from which the user can select 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.

We have included a diagram below showing the anatomy of the eye to help with understanding some of the terms used in this user guide. See Figure 1

This diagram has been included with thanks to the National Eye Institute, National Institutes of Health (NEI). The NEI website includes a valuable resource of photographs and images.

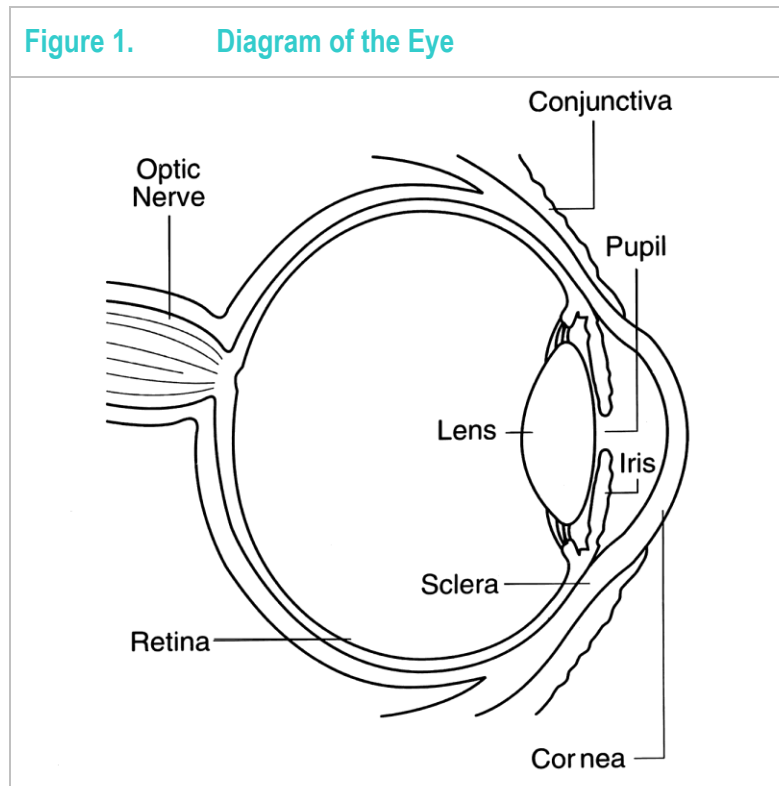


Figure 2 Shows how the *ViewPoint EyeTracker*® works in a typical head fixed configuration.

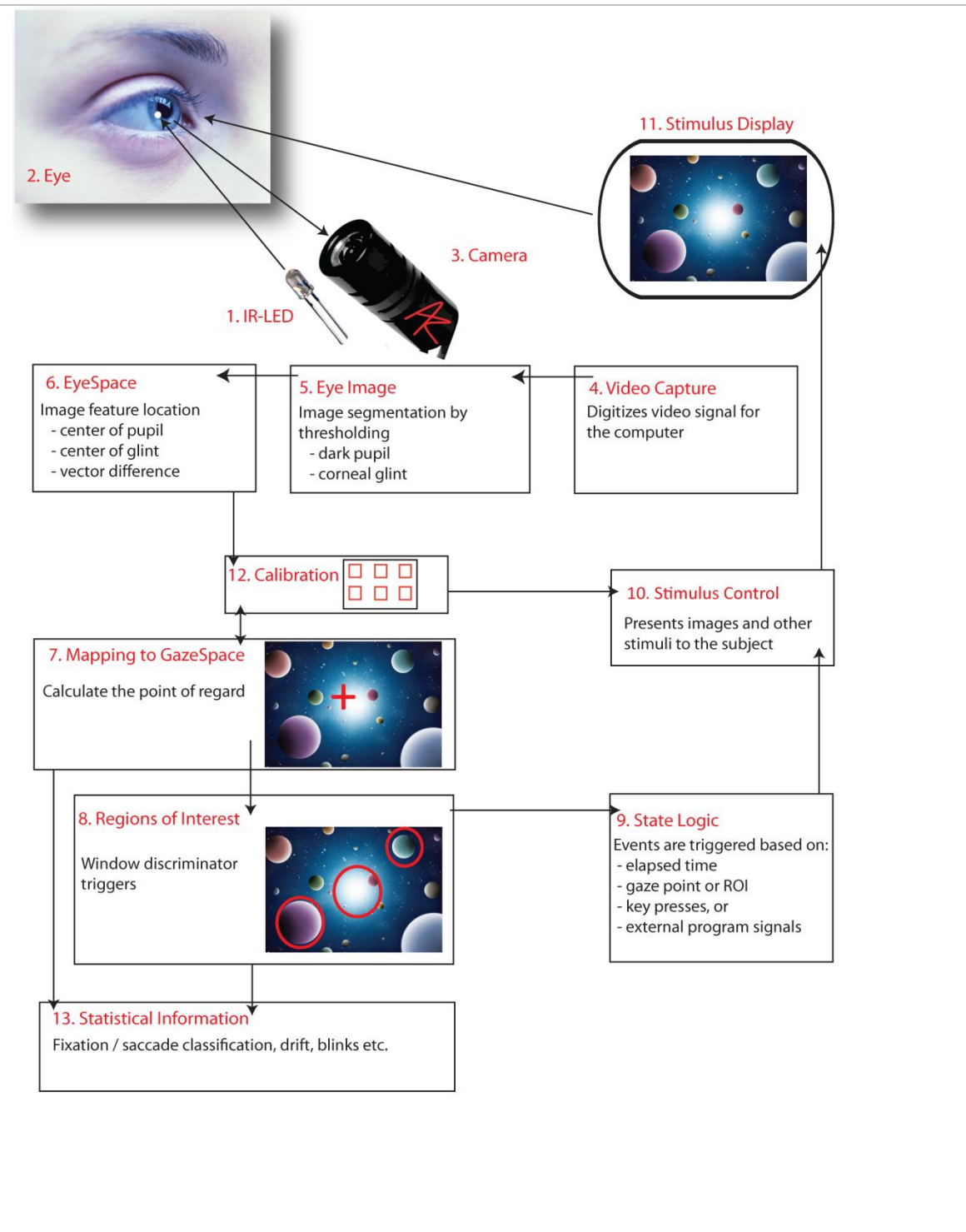
The numbers in this section refer to the item or block numbers in the figure.

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 3. 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.**).

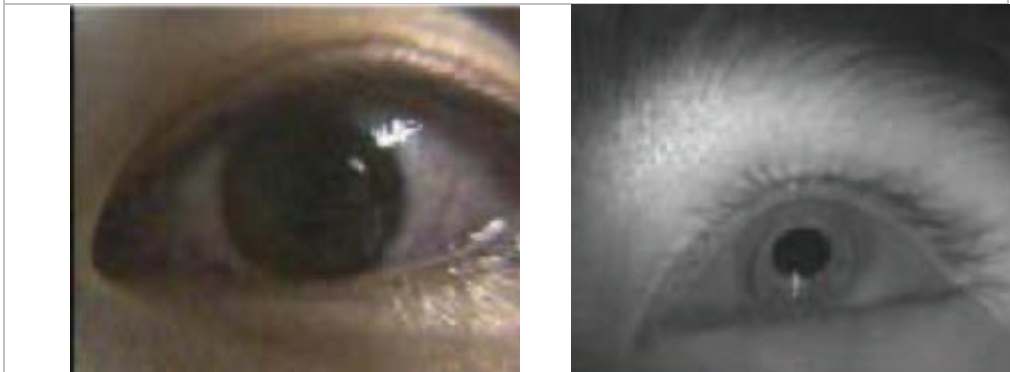
Figure 2. Schematic of the ViewPoint EyeTracker® System (head fixed)



2.2 Infrared Light

The value of using infrared light is illustrated in [Figure 3](#). 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.

Figure 3. Infrared light allows for pupil discrimination



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 Interfaces

There are many ways to interface to the *ViewPoint EyeTracker*® for data synchronization, communication and control:

- Graphical User Interface (GUI)
- Command Line Interface (CLI)
- Software Developers Kit (SDK)
- *ViewPointClient*™
- Third Party Interfaces

2.4.1 Graphical User Interface (GUI)

The most common controls are exposed as buttons, sliders, menu items etc. Please be aware that these are only a subset of the complete set of features available.

2.4.2 Command Line Interface (CLI)

The *ViewPoint EyeTracker*® provides a *Command Line Interface* (CLI) that allows users to control almost every aspect of the program. This consists of a set of *Instructions* and the *Command Line Parser* (CLP) that interprets the instructions. Each *Instruction* begins with a *KeyTerm* and may be followed by one or more *Arguments*. Command lines may contain successive instructions separated by semicolons and grouped by braces.

The following is a list of CLI *operators* and their definition. Operators follow a strict precedence which defines the evaluation order of expressions containing these operators.

Table 2. CLI Operators		
Operator	Description	Usage
EOI	End of line (OS specific)	Command terminator
"	double-quote character.	Open and close a <i>string</i> argument.
//	Double forward-slash	Comment identifier.
{ }	Open and Close braces	Begin and End (deferred) execution block
;	semicolon	Command separator
' '\t' ;	Space, Tab, Comma	Argument delimiters
:	colon	target specifier, left-to-right associativity, (no white spaces before the colon)

2.4.3 Example of a command line:

```
EyeA:autothreshold ; fkey_cmd 5 { settingsFile_Load "key 5 .txt" ; Both:snap&Inc }
```

The above line first starts the autothreshold process running for EyeA, then assigns the expression within braces to key F5. The expression consists of two instructions that are only evaluated when the F5 key is pressed, that is, their evaluation is deferred.

- **Strings** are defined by enclosing a set of characters within quotes (“”). A string must start with a quote character and end with a quote character. An error will be generated if the CLI finds an unmatched quote. Quoted strings have highest precedence, meaning that all text, including operators, inside the quoted string will be shielded from evaluation.
- **Comments** are text that is not evaluated; they are identified by a *comment identifier*, typically two forward slashes (//). A comment identifier tells the CLP that everything following on that line is to be ignored. Comment identifiers can appear anywhere on the line. Note that quotes have precedence, so double slashes inside a quoted string will not be interpreted as as a comment identifiers.
- **Braces** provide grouping of a sequence of commands, as shown in the above example where the group of instructions is assigned to an fkey. This is typically done when execution of the group deferred. Braces can be nested. An error will be generated if the CLP finds unmatched braces.
- The **colon** is used as part of a target specifier. For example, when an instruction could be applied to one or both of eyes, it is used to specify which eye, for example:
`EyeA:autothreshold`. White spaces are not allowed before the colon.
- **Delimiters** separate command arguments. They include white spaces (space-bar and tab characters) and commas.
- **Semicolons** can be used to separate multiple commands on a single line.

2.4.4 Important CLI Changes from Previous Versions

There are several important and significant changes from previous versions that may require settingsFiles to be modified.

1. Strings must be delimited by both a beginning and an ending quote. Previously, some commands allowed only a beginning quote.
2. Only real text strings such as file names should be put in quotes. Previously, deferred command strings were put in quotes.
3. Deferred commands and command sequences should be put inside matching braces.
4. Braces may be nested.
5. Commands separated by semicolons are always evaluated in left-to-right order.
6. Prefix eye target notation (e.g.: EyeB:autothreshold) is the only specification recognized. Previously, some commands allowed eye target specifictaion as an optional first argument.

2.4.5 Arguments

Some commands take arguments. Valid arguments include:

- **Boolean** : can be one of the following : `yes`, `no`, `true`, `false`, `on`, `off`, `1`, `0`, `toggle`.
- **Integer** : must be numeric digits, can include { + - }; e.g.: `-254`. Note that the negative sign can also be used to specify the direction of change on TTL lines, so `-0` indicates a voltage fall on channel 0 and `+0` indicates a voltage rise on channel 0.
- **Float** : must be numeric digits, can include { + - . }; e.g.: `0.75`
- **String** : a set of printable ASCII characters inside quotes; e.g.: `"This//,that),the :other"`

2.4.6 Asynchronous Operations

Note that some instructions (e.g. `autothreshold`, `autocalibrate`) start asynchronous operations and return before the operation has completed. Incorrect assumptions about sequential execution of instructions may lead to errors that are difficult to debug.

2.4.7 Error detection and reporting

Errors are only detected and reported at the time of evaluation. There is no detection of invalid commands at the time of deferred command assignment; e.g.: when an `fkey_cmd` is assigned.

File name arguments are entered as strings. The validity of file names and folder paths is tested only when the file name is used.

Command arguments should not be in braces unless they are deferred commands. For example, `fkey_cmd 5 { settingsFile_Load "my file .txt" }` is valid, but `penColor { 0 0 255 }` is invalid.

2.4.8 Parameters and Arguments

The terms '*parameter*' and '*argument*' are typically used interchangeably in everyday language. Technically, the parameters is the variables used when defining a command, e.g.: `penColor redVal greenVal blueVal`, while arguments are the specific values passed when calling the command, e.g.: `penColor 0 0 120`.

2.4.9 Known Problems and Issues

Both: `snap&Inc` does not currently work as expected, rather relies on modal settings.

EyeB: `PenColor` does not currently work as expected, rather defaults to EyeA, can also use `PenColorB`

These CLI strings may also be associated with F-keys for the user's convenience and also with TTL inputs. 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 interface (CLI).

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 CLI s 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 17ViewPoint Interface: GUI, SDK, CLI to see if a CLI exists.

2.4.10 Software Developer’s Kit (SDK)

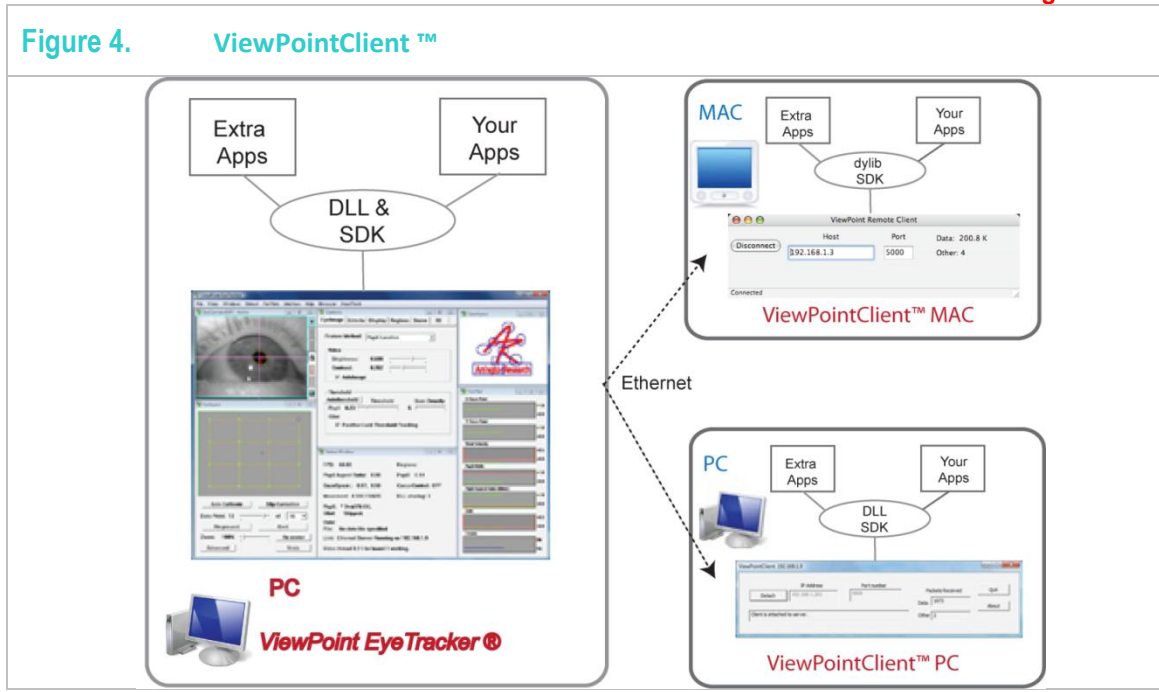
ViewPoint software includes a powerful software developer’s kit (SDK) that allows seamless interfacing with *ViewPoint* in real-time, giving real-time access to all *ViewPoint* data. It provides for calibration stimuli in the user’s stimulus window, or the user’s application to draw into *ViewPoint*’s Stimulus and GazeSpace windows. Allowing complete external control of the *ViewPoint EyeTracker*®, The SDK interface is based on shared memory in a dynamic-link library (DLL). The SDK is event / message driven so there is no CPU load from polling and provides microsecond latency.

2.4.11 ViewPoint Client™

ViewPointClient™ is a program that runs on a remote computer and communicates with the *ViewPoint EyeTracker*. It interfaces to a copy of the dll and exchanges data just like *ViewPoint* does, but typically takes less than one percent of CPU resources. This means that the same “layered” applications can be used on a remote computer just as easily as on the same computer. *ViewPoint* includes an Ethernet server; the *ViewPointClient* establishes an Ethernet link with this server. *ViewPointClient*™ for PC and MAC OSX is included free with the *ViewPoint*.

There are three auxiliary programs to help take care of inter-computer communication:

Table 3. Inter-Computer Communication	
ViewPoint Program	Function
ViewPointClient™	Ethernet communication between two windows computers. Chapter 14
ViewPointClient (MAC)™	Ethernet communication between a windows PC running the <i>ViewPoint EyeTracker</i> ®. and a MAC. Chapter 16
RemoteLink™	Serial communication between a windows PC running the <i>ViewPoint EyeTracker</i> ® and either a Windows PC or a MAC. May not be supported in the future.



2.5 Interfaces to Third Party Products – Integrity Suite™

Integrity™ is a suite of interfaces between the *ViewPoint EyeTracker®* and 3rd party applications, which is provided by *Arington Research, Inc.* to ensure a professional quality, uniform, complete, and thorough interface to the favorite products of eye tracker users. These provide access to data, complete eye tracker control, and data integration and synchronization, all in real-time. *Integrity* is included free with *ViewPoint*.

Interfaces include:

- ViewPoint EyeTracker® toolbox for MATLAB®
- EBasic interface for E-Prime®
- LabView
- Python
- Presentation®

2.6 Others

Many third party applications provide the means to load our DLL into their program which means that a user can call ViewPoint SDK functions from within that application. These include MATLAB®, LabView and Python.

2.7 Sample Interface Applications

Run the sample *layered* applications to demonstrate communication with *ViewPoint* either directly via the DLL on the same machine, or indirectly via the DLL and one of the auxiliary programs,



running on a second machine. Simply, copy the sample interface applications from folder **ExtraApps** to the folder containing *RemoteLink* and the dll on the second machine. These may include:




- **DataMarker.exe** – Allows easy manual insertion of data file marker characters into an open *ViewPoint* data file.
- **VPX_MFC_Demo.exe** – All **Data File** group controls will work from the remote machine. Also, Stop *ViewPoint* button will work, but it is impossible to launch the *ViewPoint EyeTracker* remotely, since *RemoteLink* must already be connected to that application (rather than to the local DLL).
- **VPX_Win32_Demo.exe** – All other controls that send one command at a time will work from the remote machine, except for the launch commands, as discussed above. We are working on resolving this problem.
- **VPX_Basic_Demo.exe** - opens, pauses, resumes and closes data files, displays streaming x and y position data and also includes a means to send CLI commands.

2.8 About Writing Layered Applications

Layered applications communicate seamlessly with the *ViewPoint EyeTracker* via the **VPX_InterApp.dll** nexus. Separate documentation and examples are provided in the folder **ViewPoint/SDK**. *ViewPoint* users are encouraged to write and share their own layered applications with the user community. We will be more than happy to help

Chapter 3. QuickStart Guide

It is essential that you read this guide before attempting to collect any data using your *ViewPoint EyeTracker*. Using the incorrect calibration procedure will give you unusable data.

System	 <p style="text-align: center;">SceneCamera Systems</p>	 <p style="text-align: center;">Head Fixed Systems</p>	 <p style="text-align: center;">Head Mounted Display (HMD Systems)</p>
Description	<p>The SceneCamera option allows ViewPoint EyeTracker® users to track gaze position on a real world scene video. No head tracker is required to obtain eye position as the calibration is with respect to the scene camera which moves with the subject.</p> <p>The subject is free to move around.</p>	<p>The ViewPoint EyeTracker head fixed systems are perfect for visual psychophysics, EEG studies etc. that require a stable viewing position or restriction of head movement. Calibration is performed with respect to a computer screen or projector display.</p> <p>The subject's head must remain fixed during the entire duration of calibration and data collection.</p> <p><i>Procedures for the close focus camera setup or desk mounted camera system are the same.</i></p>	<p>Calibration is performed with respect to the HMD display.</p> <p>The subject is free to move around.</p>
Calibration	<p>Calibration is performed relative to the pixels of the CCD array, NOT the image content. This is analogous to calibrating relative to the CRT screen and NOT the image displayed on it.</p> <p>Use the Re-Present feature to individually calibrate to the scene image.</p> <p>Refer to section 4.4.2 in the User Guide</p>	<p>Automatic on-screen calibration.</p> <p>Refer to section 4.4.1 in the User Guide</p>	<p>Automatic on-screen calibration.</p> <p>Refer to section 4.4.1 in the User Guide</p>
Output	<p>AVI 2.0 movie output from the scene camera showing position of gaze painted on it. Corresponding ASCII data file.</p>	<p>ASCII format files showing position of gaze relative to the screen calibrated over.</p>	<p>ASCII format files showing position of gaze relative to the HMD screen.</p>

It is extremely important that you read this user guide and understand it. Initially Chapters 4, 5, 6, 9 10 and 11 will get you started. Do not wait for your subjects to be ready. Calibrate and collect data on yourself first. Always collect and analyze pilot data before running your experiments so that you understand exactly what you are getting.

Chapter 4. Installation and Setup PC-60

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).

4.1 Computer System Requirements

This is the installation guide for *ViewPoint PC60* that runs on the Windows XP, Vista and Windows 7 operating systems. As of version 2.8.4, we no longer officially support Windows NT, Windows 98, Windows ME, or Windows 2000. Windows 2000 may work if the latest service packs are loaded, however we do not support this.

We do not currently support 64 BIT operating systems.

* 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.

4.2 Video Requirements

ViewPoint EyeTracker[®] systems 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.

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

4.3 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. To change the video input type for the EyeCamer, the user should use the monitor icon on the EyeCamera window > **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 SceneCamera video standard (signal standard) can now be changed using the following command:

```
scene_videoStandard <format>  
where <format> is one of: NTSC, PAL, SECAM
```

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

4.4 PCI Video Capture Card and Driver Installation

Important: If you 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.

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.

Installing the New Driver

Important: Disconnect from the any network or internet prior to installing the drivers.

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.

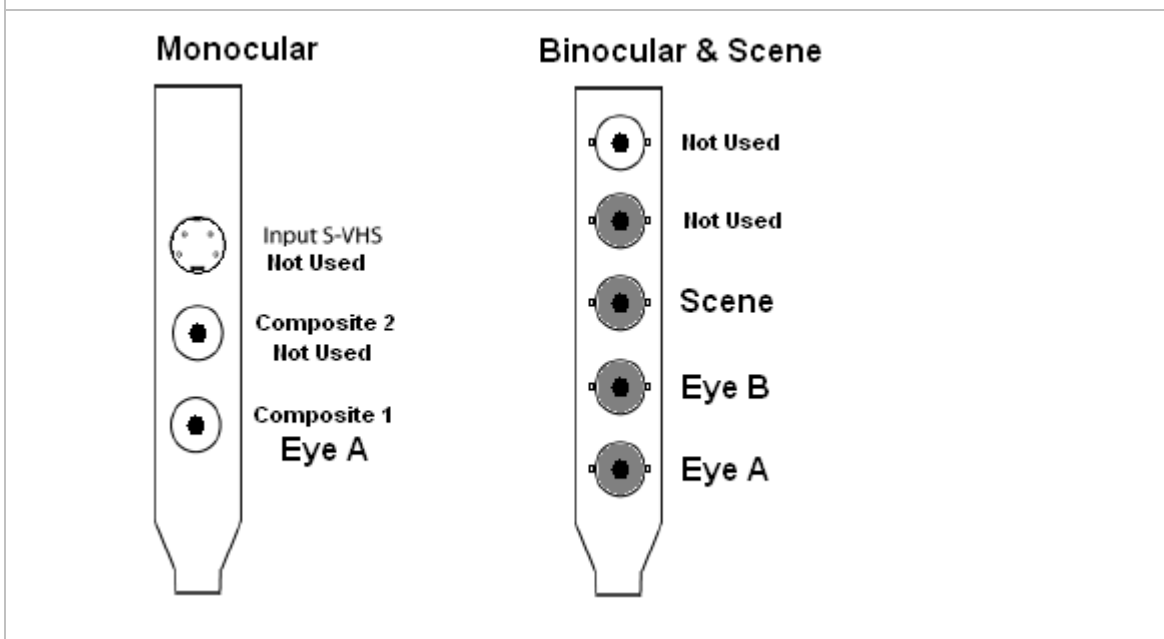
Note 3: The video cable must be connected to the composite input number 1 of the frame grabber for monocular ViewPoint. Refer to Figure 4

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

Note 5: If a driver installation problem occurs, please see the device manager under sound, video and games controllers. Devices that are not correctly installed will usually have an exclamation mark next to them. If you see any video inputs marked as “KWorld” devices, please contact us for a solution.

The *ViewPoint EyeTracker*® is delivered with one of the following PCI cards, depending upon the product that was ordered. The video channels are statically set as follows:

Figure 5. Frame Grabber Connections



Various cable options are available, but in general we try to adhere to the following color codes:

Yellow ● -----Eye_A Camera

Red ● -----Eye_B Camera

White ○ -----SceneCamera

Black ● -----Power (12 VDC, center positive)

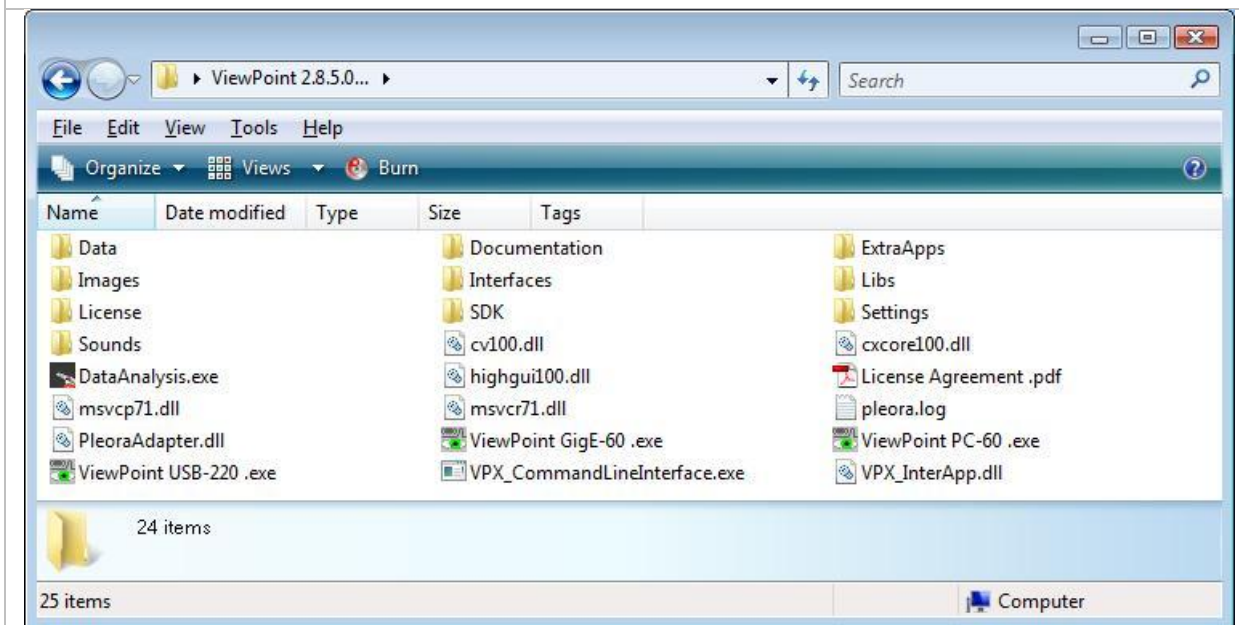
4.5 ViewPoint EyeTracker® Software Installation

Copy the **ViewPoint** folder from the CDROM to the hard drive of your computer. This folder is illustrated in [Figure 6](#). This directory structure must be maintained for proper functioning of the software. The *ViewPoint EyeTracker*® 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.

Notes:

1. if you purchased a PC-60 system then click on the ViewPointPC-60.exe, USB 220 users click on the Viewpoint USB-220.exe and GigE-60 users the Viewpoint GigE-60.exe.
2. on Windows Vista you must run the application as an administrator to be able to use any of the movie playing and recording features.

Figure 6. The ViewPoint EyeTracker® Folder



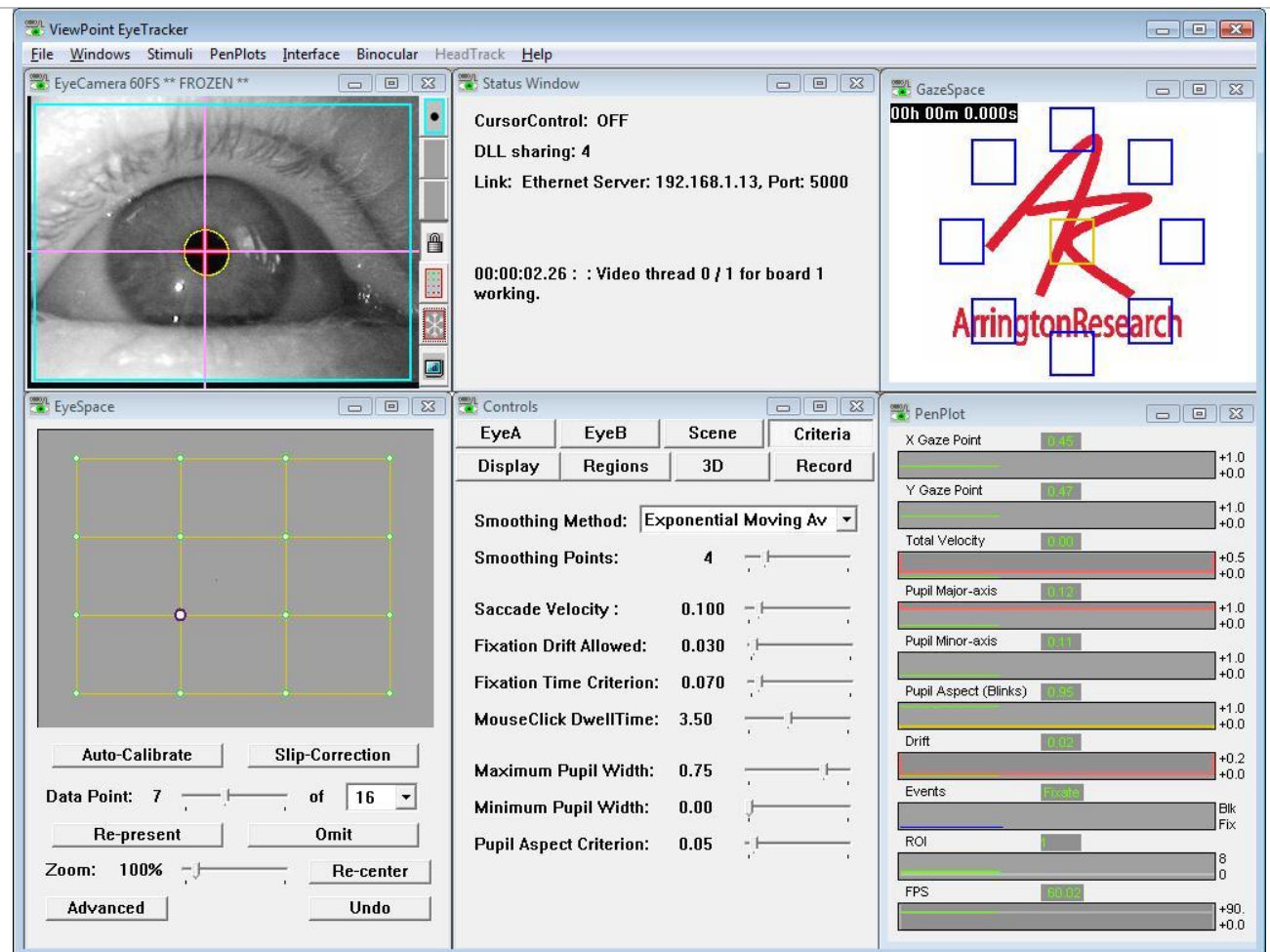
4.6 ViewPoint License (.VPL) File

The video capture board, IP Engine or USB camera (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.

4.7 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



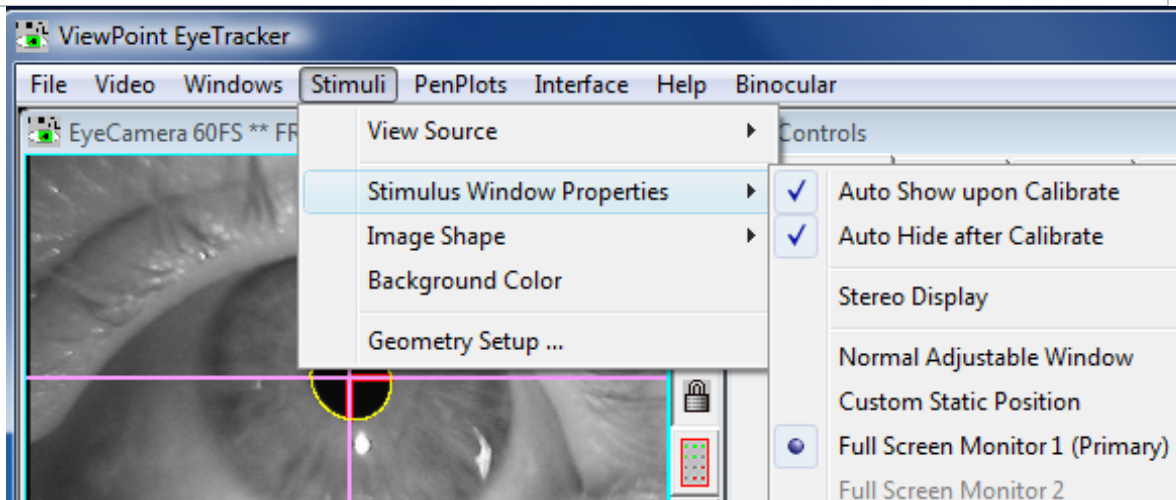
Note: The normal window layout requires a display at least 1024 x 768. An extended layout is also provided for display screens with a resolution of at least 1280 x 1024. Select menu item: *Windows > Arrange > Larger Layout*

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.

4.8 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 8. 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
Windows >	Open and close windows, specify window layout.
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 Ethernet server or serial port interface.
Binocular >	Binocular EyeMovement Settings. Dimmed if Binocular Option not purchased.
Help >	Access system configuration information, View license Holder information, Access the <i>Documentation</i> folder, and Link to the Arrington Research, Inc. web site

Table 4. Window function descriptions	
Window	Function:
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 recalibration of individual points or the ability to omit problem points. Refer to Chapter 7
EyeSpace	Miniature representation of the stimulus window to enable experimenter to monitor eye movement during recording.
Controls:	<p>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.</p> <p>Eye tab: Eye Image quality adjustments and tracking method specification.</p> <p>Criteria tab: Specify smoothing and other criteria to apply to the data.</p> <p>Display tab: Specify information to be displayed in the Stimulus and GazeSpace windows. Also to specify information to be displayed in recorded SceneMovies for users with the SceneCamera option.</p> <p>Regions tab: Setup regions of interest (ROIs) and calibration regions.</p> <p>Scene tab: Adjust brightness, contrast, hue, saturation etc. of scene image (scene camera option only).</p> <p>3D tab: only for 3D-WorkSpace Option</p> <p>Record tab: Quick and easy way to open, pause and close data files as well as insert markers.</p>
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 10
Penplot:	Real-time plots of, for example: 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.

4.9 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.

These criteria can be divided into two categories: Feature Criteria and Movement Criteria.

Table 5. Criteria	
Feature Criteria	Movement Criteria
<ul style="list-style-type: none"> • Pupil minimum width • Pupil maximum width • Pupil aspect ratio 	<ul style="list-style-type: none"> • Drift • Saccade velocity threshold • Fixation minimum duration

The new user should be able to proceed through the Quick Start sections in this manual without needing to adjust these criteria. However, it is important to understand them prior to undertaking data collection.

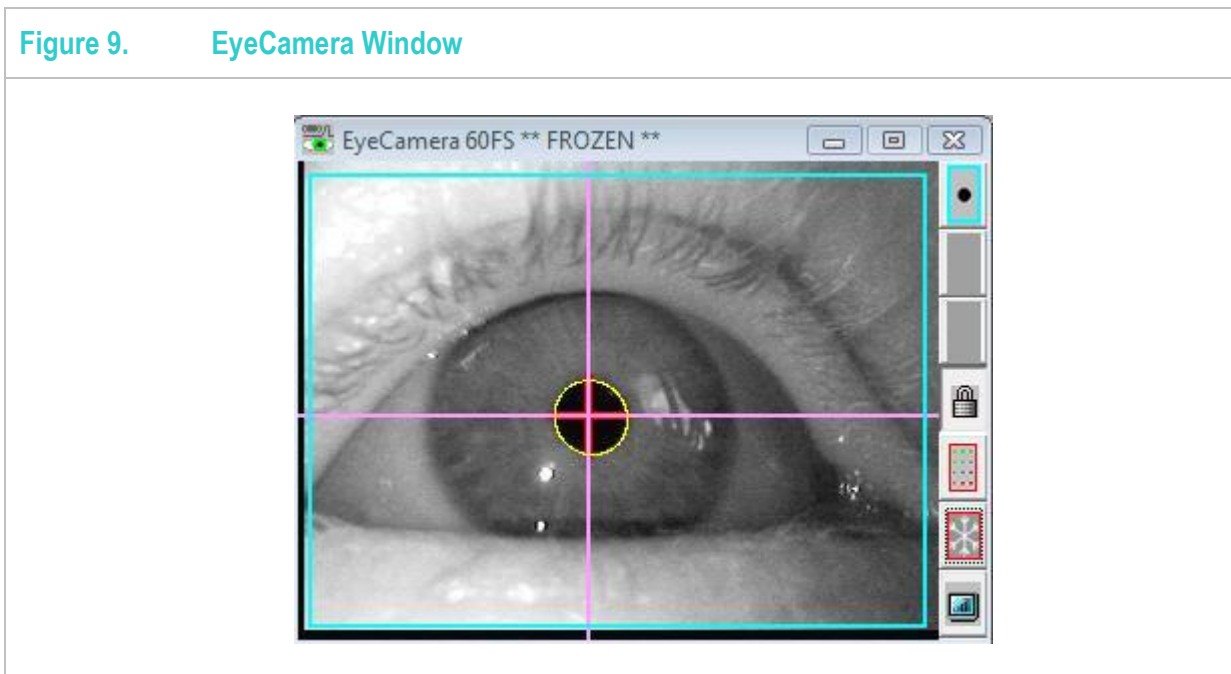
Chapter 5. Quick Start Section

This chapter contains all of the information for a new user to get up and running with the system 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. Refer to Chapter 6

5.1 Camera Positioning

This section describes correct camera positioning. Refer to the correct set of instructions for your system. The object in all cases is to obtain an image of the eye similar to that below in Figure 9

Figure 9. EyeCamera Window



5.1.1 QuickClamp Lipstick Style Camera Positioning

Instructions for proper positioning of the LipStick style Camera & LED when using the QuickClamp hardware.

1. Position the camera at 45 degrees below the line of sight of the subject (as they are viewed from the side), approximately 4-7cm from the eye.
2. Position the LED so that it appears at the 11 o'clock position for the right eye and 2 o'clock for the left eye, 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.



3. 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. Movement between the head and the EyeCamera must be minimized, which is the purpose of the *Arington Research Precision Head Positioner* and camera system.

5.1.2 EyeFrame Camera Positioning

1. Place the EyeFrame glasses on the subject.
2. Use the collar clip to reduce cable weight and twisting.
3. Tighten the strap to minimize movement on the head.
4. Adjust the position of the EyeCamera to obtain an image of the eye where the pupil is in the center of the EyeCamera window when the subject is looking straight ahead, and the eyeball fills the maximum possible area as shown in Figure 7.
5. 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. The camera may be closer in if the gaze is not eccentric.
6. The lenses on the EyeFrame eye camera can often be rotated manually by pinching the part of the lens extending from the housing. Otherwise use the two holes on the lens front surface.
7. 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.

5.1.3 Remote System Camera Positioning

As remote systems are sold to accommodate varying operating setups, exact positioning of the camera and illuminator will also vary. Position both the camera and illuminator to achieve a well illuminated image of the eye that fills the whole EyeCamera window.

5.2 Locating the pupil – all systems

The following steps are applicable to all systems.

- a. 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 9
- b. 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.

- c. 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.

ViewPoint systems are designed to produce a uniform diffuse IR illumination and you will not see this effect.

- d. 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.
- e. 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.

5.2.1 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.

5.3 Thresholding – all systems

For simplicity and ease of use, keep the **AutoImage** and **Positive Lock threshold** tracking options checked. If you do need to adjust these manually, refer to Chapter 6.

Note that the pupil scan area must be at least 50% of the EyeCamera image area for the AutoImage to function well.

Note: If your subject has a small pupil then you need to adjust the Scan Density to 5

5.4 Calibration

Refer to the section for your particular system.

Try to use at least 9 calibration points, 16 points usually provides very good calibration.

It is good practice to integrate slip-correction into your experiments. Choose a good center point and perform the slip correction at the working distance.

5.4.1 Calibration (Head Fixed and HMD Systems)

For the HeadLock System, QuickClamp, Remote system, or other systems where you are fixing the head, 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 10](#).

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

1. Warn the subject of the onset of the calibration stimuli to ensure successful calibration.
2. Instruct the subject to look directly at the center of each stimulus until it converges to a point. The default is for the calibration stimulus points to appear in random order.
3. Start the calibration by pressing the **Auto-Calibrate** button on the **EyeSpace** window. The warning 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.
4. 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.
5. 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.
6. Stray calibration data points can be identified and re-calibrated or omitted. 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.
7. Select the stray calibration point by left mouse clicking the point in the **EyeSpace** window.
8. 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 warning 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.

9. If a particular point cannot be recalibrated, then select that point and press the **Omit** button.
10. 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.

5.4.2 Calibration (scene camera)

Calibration is performed relative to the pixels of the CCD array, NOT the image content. This is analogous to calibrating relative to the CRT screen and NOT the image displayed on it.

After ensuring successful eye image thresholding, perform the following calibration steps to provide position of gaze with respect to the scene camera image.

1. Select menu item: Stimuli > View Source > Scene Camera. This will cause the scene camera to be displayed in the GazeSpace window. It also changes the calibration mode to snap and increment (refer to EyeSpace window advanced settings) and causes the calibration stimulus point array to be displayed in the GazeSpace window.
2. Adjust the scene camera so that the center of the subjects straight ahead field of view is at the center of the image.
3. If necessary, adjust the brightness and contrast of the scene camera image using the controls window scene tab. Reduce brightness such that overlays are visible.
4. Ask the subject to stand or sit comfortably and to remain still for the duration of the calibration process. They should not look at the computer screen.
5. Position a pen or your finger in front of the subject so that the tip appears inside the **GazeSpace** window in the active (blue) calibration circle.

Note: For best accuracy try to calibrate at around the viewing distance that the subject will typically be working at.

6. Ask the subject to move their eyes (keeping their head still so that the pen remains in the active calibration point) to look at the tip of your pen.
7. When you are sure that they are looking at the point (ask for verbal confirmation) select the **Re-Present** button on the **EyeSpace** window. The asterisk on this button indicates “snap” presentation mode and the ++ indicates “Auto-increment”. The F8 key can be used as a shortcut key for this button (this is the default FKey command).
8. Repeat steps 5-7 with each active calibration point until you have completed the set.
9. Refer to 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. Uniform curvature of the field of dots is acceptable.
10. Stray calibration data points can be identified and re-calibrated. Select the stray calibration point by left mouse clicking the point in the **EyeSpace** window. Alternatively 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.
 11. Repeat steps 5-7 for the stray calibration point. 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.
 12. 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.
 13. A consistent offset in position can be corrected using the slip-correction feature. Select a calibration data point in the middle of the group. Repeat steps 5-7 above but press the Slip-Correction button instead of the re-Present button. This automatically adjusts the calibration to compensate for the measured slip in the X and Y planes.

The line of sight of the scene-camera is not exactly the same as that of the eye. This will cause a slight constant offset if the calibration point distance is different from the distance of the gaze point. The Binocular version of the scene camera system includes automatic and manual correction for this parallax error.

5.5 Stimuli

5.5.1 Choosing the Type of Stimulus

The ViewPoint EyeTracker can be used with a variety of types of visual stimulus environments, including (a) stimulus images presented in the ViewPoint Stimulus Window, (b) real world environments as would be viewed while walking about captured by the EyeFrame SceneCamera, or (c) interactive work on a computer display captured as a movie of screen snapshots. See menu item: **Stimuli > View Source >**

You can also interface to third party applications e.g. stimulus presentation and experimental control programs, and virtual reality applications; however these are beyond the scope of this Quick Start Section. See Interfaces folder in the ViewPoint folder.

The following sections describe setting up what stimulus environments and saving data in these environments.

5.5.2 ViewPoint Stimulus Window

Using ViewPoint in head fixed mode or with an HMD, you can present stimuli in the ViewPoint Stimulus window as follows:

1. Present static BMP images and sequences of BMP images in the ViewPoint stimulus window. The position of gaze over the images is recorded Select menu item: **Stimuli > View Source > ViewPoint stimulus window.**

2. Stimulus movies: Future capability.

5.5.3 Interactive Computer Display

Screen Movies: the subject's position of gaze is recorded as they interact with the computer display screen. A movie is recorded that can then be played back showing the position of gaze as the subject opens and closes windows etc. Select menu item: **Stimuli > View Source > Interactive Computer Display**.

5.5.4 Scene Camera systems

When the scene camera option is enabled, the user can select menu item: **Stimuli > View Source > Head Mounted Scene Camera** The stimulus is then the real world scene as the subject moves freely around. If the SceneCamera option is enabled, whenever a ViewPoint DataFile is recorded, a SceneMovie will also be created. There are several options for the SceneMovie, including format, selection of overlay graphics, compression. ViewPoint synchronizes the SceneMovie with the DataFile by asynchronously inserting a TimeStamped frame number every time a movie frame is stored. This allows retrieving the corresponding SceneCamera frame image as the data from the DataFile is read. Several eye tracker data values can also stored inside the movie frame (currently this is limited to uncompressed movies). This allows the eye tracking data to be later extracted from the SceneMovie without reference to the DataFile.

5.6 Data Collection – All Systems

Now that the system is calibrated, data can be collected.

PC-60 and GigE-60 systems: You can select the required sampling rate using the monitor icon on the EyeCamera Window. This will bring raise a series of menu items. Select Menu Item Mode >. (**SetUp – High Precision – High Speed**). *Important:* Be sure to let the video mode stabilize for a few seconds, before clicking the button again. Refer also to Chapter 11 Data Collection.

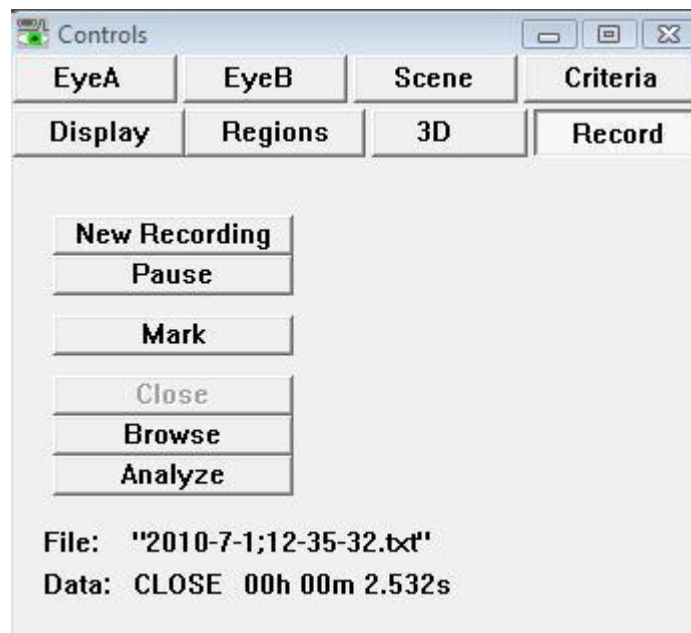
To start recording data to file: Select menu item: **File > Data > New Data File**. This will prompt you to create a new file in the directory **Data**. When data is being stored the **Status** window will show: **DATA: OPEN "datafilename.txt"**

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.

You can also use the Controls Window: Record tab for easy data collection controls. See Figure 10

Figure 10. Controls Window: Record Tab



For further information on data file formats refer to Chapter 11 Data Collection

5.7 Recording Scene and Screen Movies

5.7.1 Recorded Movie Format

The default movie format is AVI 2.0. We recommend this format be used unless there is some good reason to do otherwise. Recording length is limited only by disk space.

5.7.2 Display of Movie Data

ViewPoint provides the user with two options for way the data is saved and displayed on the scene movie. Care must be taken to choose the one most suited to your needs. Menu Item: [File > Data > Save Movie DataMode >](#)

[Painted](#) (default)

When this option is selected, all the overlay graphics selected to be shown in the GazeSpace window are also permanently painted into the frame images of the movie. This is a very convenient way to make easily distributable movies with gaze position overlays. These overlays can not be removed. The user should take care to deselect overlay graphics, such as the calibration array, if they are not wanted as a permanent part of the movie.

1. **Embedded**

When selected, several eye tracker data values are also stored inside the movie frame. This allows the eye tracking data to be later extracted from the SceneMovie without reference to the DataFile, and it allows the user to choose which data to overlay during the post-hoc analysis. Currently this is only available with no compression.

5.7.3 Compression

When recording AVI movies, the user can select to compress the movies if required. Select menu item: **File > Data > SaveMovie Compression**

1. None (default)
2. IV50

IV50 is the fourcc code for Intel Indeo version 5.10 and higher codecs. This is currently the only compression codec offered. It was chosen for several reasons. First, because it is supposed to be available on all machines (relieving the user from the ordeal of selecting a suitable codec, searching for and installing codecs and getting involved with cumbersome codec licensing issues). Second, because it provides reasonable quality images without overwhelming CPU burden.

Note: Digital movie creation and playing is not a fully mature area in computer science; as it evolves, so will with ViewPoint EyeTracker movie interface and format and compression options.

5.8 Data Analysis

5.8.1 Head fixed and HMD systems

There are many options to view and analyze the data recorded, including.

1. Real-time in the GazeSpace and PenPlot Windows.
2. Using the Data Analysis program provided free, as is, with ViewPoint. See separate PDF documentation.
3. Post-hoc analysis of the ASCII data files using Excel, MATLAB, Mathematica, etc..

5.8.2 Playing Scene and Screen Movies

AVI movies with Painted Data, either Compressed or Uncompressed, can be easily distributed, downloaded, and played with with most any movie player software, including Microsoft Media Player. This is by far the simplest and easiest, and is therefore the default.

AVI movies with or without Painted Data, either Compressed or Uncompressed, will be loaded automatically into the Data Analysis program (see separate PDF documentation) when the corresponding data file is loaded.

To play VPM movies you need to use the VPM_Viewer.exe program. This extracts the eyetracking data from the frames of the movie and provides the user with data display options.

AVI without Painted data requires the associated ViewPoint Datafile for the eye tracking data.



The ViewPoint DataAnalysis program is currently provided as a beta version, without charge and AS IS; it may be useful, but ARI provides no official support for it. (see separate PDF documentation)

5.9 Frequently Used Settings

You can create settings files and place them in the StartUp folder 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, or to have Binocular Mode automatically turned on. When *ViewPoint* is launched it loads in all of the .txt files in this folder, which can reduce setup time.

5.10 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 , place it in the StartUp Settings folder.

5.11 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 CLI action. This is done via the CLI 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 [17.24](#)

5.12 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 6. Locating the Pupil and Glint (all systems)

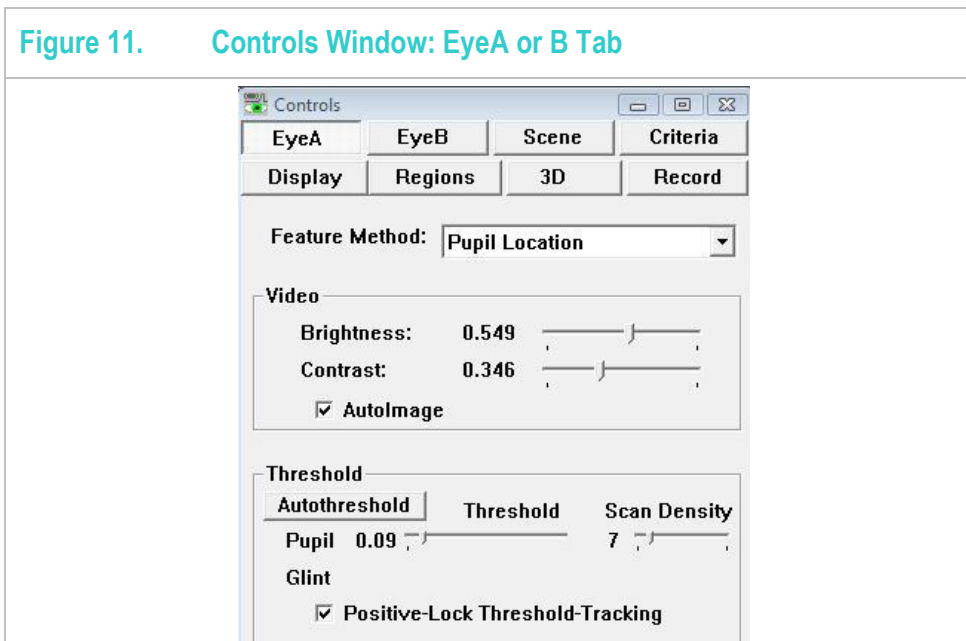
Important: The ViewPoint software comes with two very powerful features that will automatically and continually provide the best setting for tracking each subject:

- **Video AutoImage:** See the [AutoImage](#) check box located on the [Controls Window Eye](#) 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.
- **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 Eye](#) tab. Note: this only adjusts the pupil threshold; the glint threshold must be adjusted manually.

It is highly recommended that the user works first with these features before making any manual adjustments. However, there will occasionally be situations and subjects that may require some manual intervention to provide the most successful tracking.

The Controls window: EyeImage tab is shown below

Figure 11. Controls Window: EyeA or B Tab

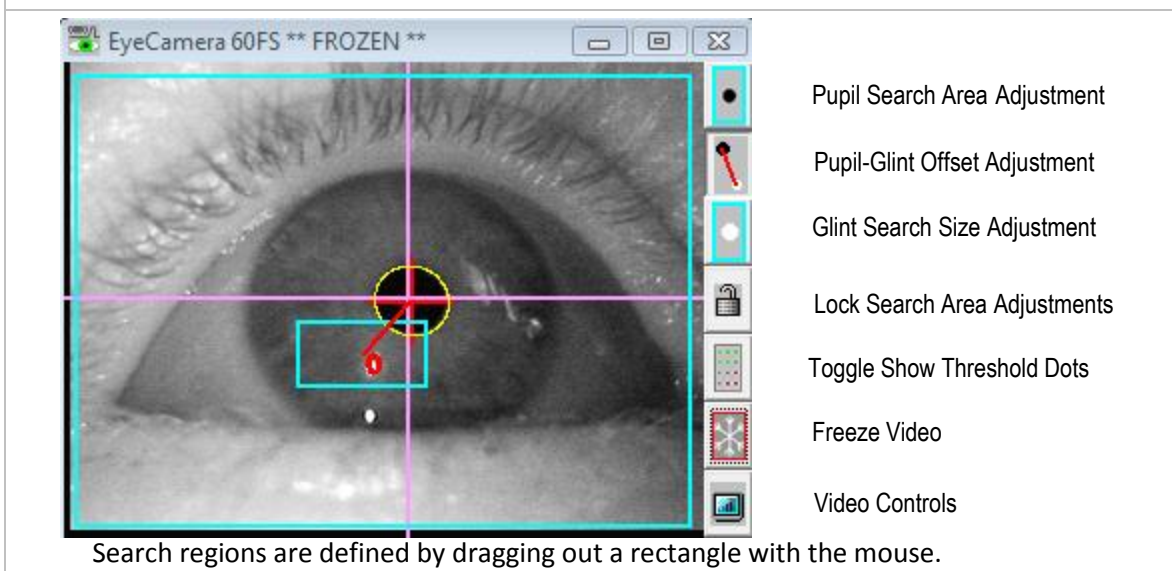


This section explains the various parameters that can be adjusted and describes when and how to make them.

6.1 EyeCamera Window

The **EyeCamera** window displays the video-image of the eye, as well as overlay-graphics that graphically provide information about image segmentation and performance. See Figure 12 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.

Figure 12. EyeCamera Window



6.2 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 user to select which features to use. This combo box is on the Eye 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 head with respect to the camera. 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 resolution and 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 6.9, and data Simulation options described in section 6.3 below.

6.3 Simulation of Gaze

6.3.1 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.*

6.3.2 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. Controls window, Video Image tab, Feature Method drop down, Pattern Simulation. (*Note this is designed to replace the option: Interface > Serial Port > Send Test Pattern, with a more general, earlier stage, simulator.*)

6.4 Thresholding

The software attempts to locate the pupil by searching for a dark region within the pupil search area. In the **Controls** window : **EyeImage** tab, **Feature Method** group, select **Pupil Location**. 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. 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.

6.5 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



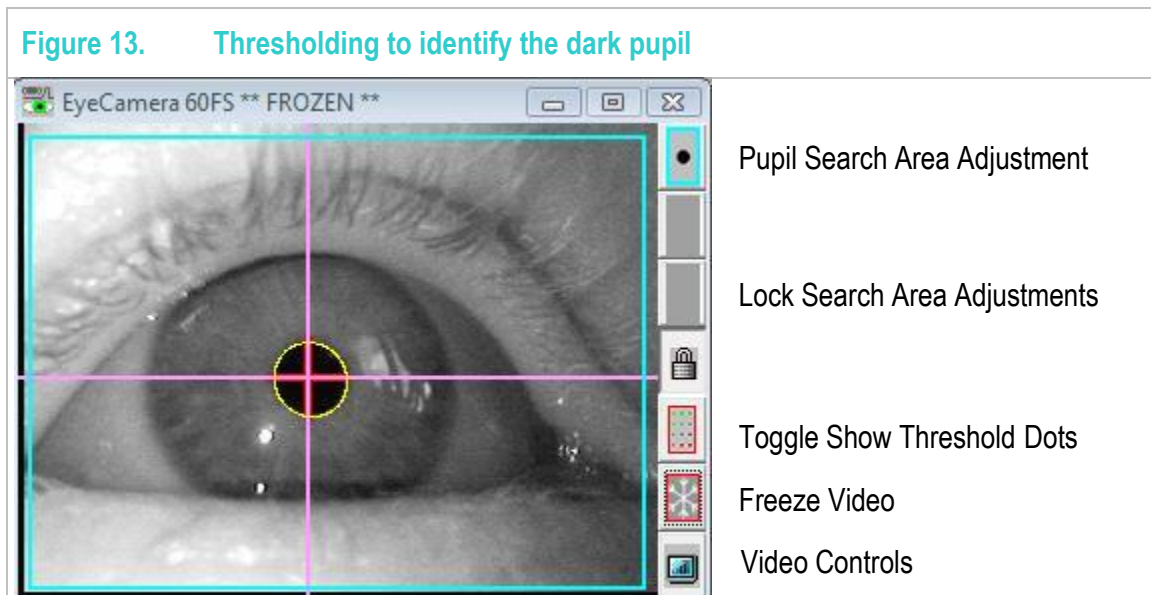
Arington Research

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](#) is toggled on.

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.

6.6 Manual Thresholding of the Dark Pupil

Follow the steps below to manually undertake the manual thresholding process for the dark pupil:



1. Toggle show threshold dots “ON” using the button in the [EyeCamera](#) window.
2. 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.
3. Select the [Pupil Search Area Adjustment](#) icon at the top right of the [EyeCamera](#) window, illustrated in [Figure 12](#). 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. 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 first imagined.
4. Press the [AutoThreshold](#) button on the [Controls](#) window: [threshold group](#).

Note: 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. 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.
6. 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.

7. 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 12](#). 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.

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. 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](#) will help in many situations.

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.

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

Follow the steps below to manually undertake the thresholding process for the Glint-Pupil vector method:

1. In the [Controls](#) window select Glint-Pupil Vector.
2. Follow the steps in 5.2 to threshold the dark pupil.
3. 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 13](#).
4. 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.

5. Select the Pupil-Glint Offset Adjustment icon on the [EyeCamera](#) window, illustrated in [Figure 12](#). Use the mouse to drag out the offset vector from the center of the pupil to the center of the corneal glint.
6. 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.
7. 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 12](#).
8. 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. 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.

6.8 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.

6.9 Automatic Slip Compensation

Some hardware configurations, for example Head Mounted Displays (HMDs) can pose a problem when the camera position constantly slips with respect to the eye. Automatic Slip Compensation 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 **Eye** tab.

Three parameters can be adjusted to specify the degree of compensation applied to the data using CLP commands. Refer to 17.7

Note: do not confuse this with calibration Slip Correction.

6.10 Feature Criteria

6.10.1 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 12 was taken with a micro camera (arranged as in [Figure 2 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.

6.10.2 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.

6.11 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 monitor icon on the **EyeCamera** window > **Pupil Segmentation Method** > *** is used to make a selection.

6.11.1 Ellipse

This provides a general rotated ellipse, so that a good fit is made to oblique pupil images. It requires 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 than with the Oval Fit method, however the pupil center calculations is usually more accurate, because more points are used for fitting.

6.11.2 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. .

6.11.3 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.

6.11.4 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.

6.11.5 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. The monitor icon on the [EyeCamera](#) window > [Glint Segmentation Method](#) > *** is used to make a selection.

6.11.6 Pupil Scan Area Shape Options

The user can specify whether to change the scan area for the pupil to either rectangular or elliptical using CLI s. Elliptical scan area can be effective at eliminating dark spots that the software may interpret as a pupil. Refer to [17.6](#).

Chapter 7. Calibration

This section describes the calibration process in more detail and also many of the advanced calibration features available to the user.

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

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. 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*).

7.1 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 § 7.7).

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.

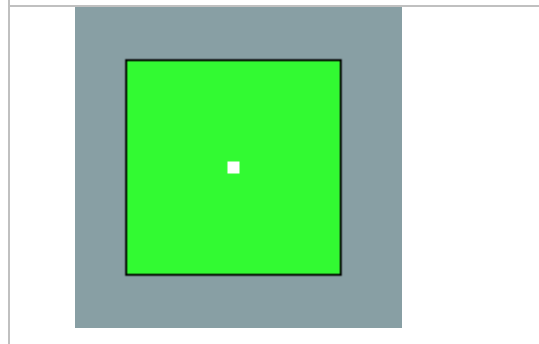
7.2 Choosing the Number of Calibration Points

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.

7.3 Automatic Calibration (Head Fixed)

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.

Figure 14. Calibration Stimuli



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](#)).

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. Speed of presentation of the calibration stimulus points can be 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.

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.

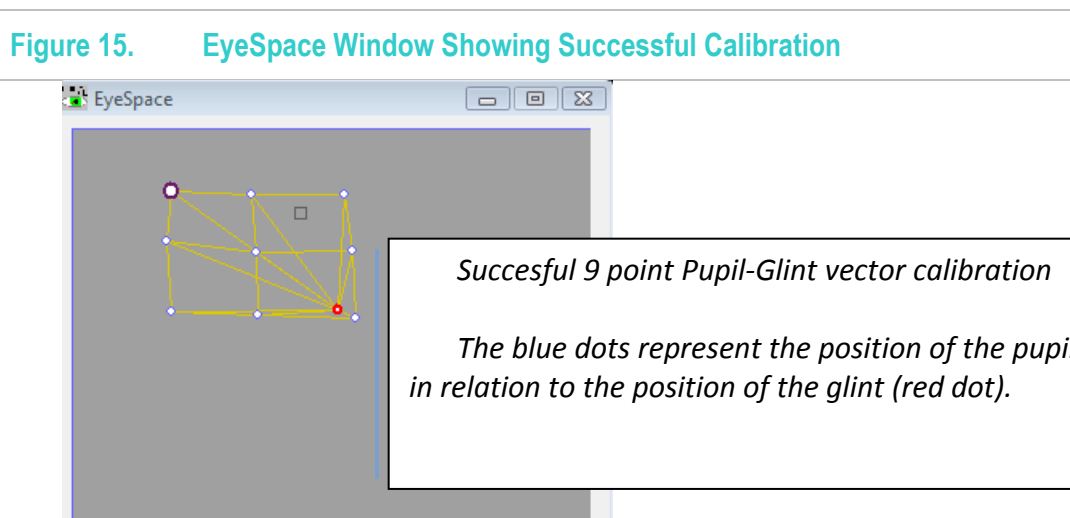
The calibration stimulus presentation order type may be changed. Press the [Advanced](#) button in the [EyeSpace](#) window for access to the controls. See section 7.11.1.

7.4 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).



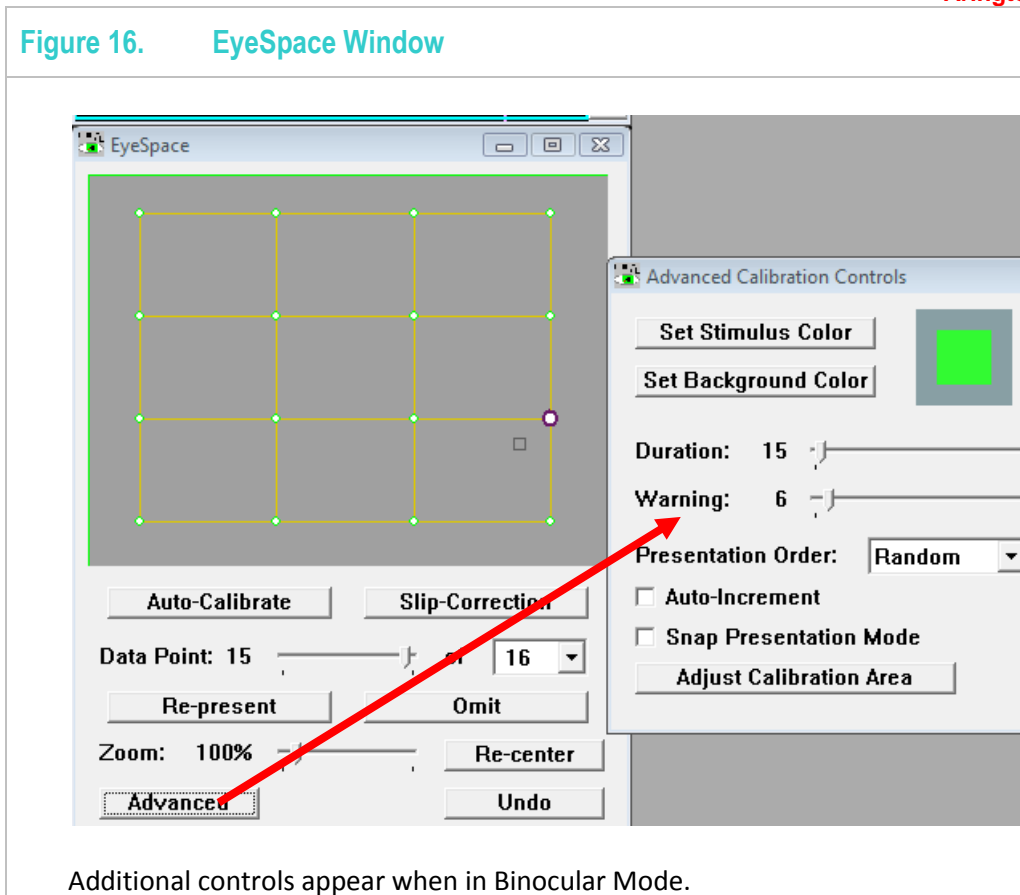
The top part of the window shows a 320x240 graphics well that represents of the coordinate space of the [EyeCamera](#) window. If [Pupil Location](#) or [Glint Location](#) modes are selected, a black square shows the current pupil location, or the current glint location, respectively. In [Glint-Pupil 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.

7.5 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. When in binocular mode, calibration points can be omitted for either or both eyes.

Figure 16. EyeSpace Window



7.6 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 13.6.

7.7 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 13.6.

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 6.9.

7.8 Parallax Correction for Binocular Scene Camera systems

Note that this section is only applicable to Binocular SceneCamera systems.

Because the optical axis of the SceneCamera is different from the optical axis of the EyeBall, a parallax error will occur in the calculated position of gaze (POG) when the subject is looking at points in a plane that is closer or farther away from the plane of calibration. The binocular SceneCamera system provides a built-in default parallax correction and also the ability to fine tune the correction.

The parallax correction value is a linear function of the binocular vergence angle. The slope of the line depends on the inter-pupillary distance (IPD) of the eyes and the distance of the SceneCamera above the line between the two eyes. In general, the default value offers substantial correction value, based on the typical camera location on the *ViewPoint EyeTracker*® EyeFrame™ and typical IPDs .

To show the position of gaze with the Parallax correction, select **Menu Item: Binocular > Show average with parallax correction**.

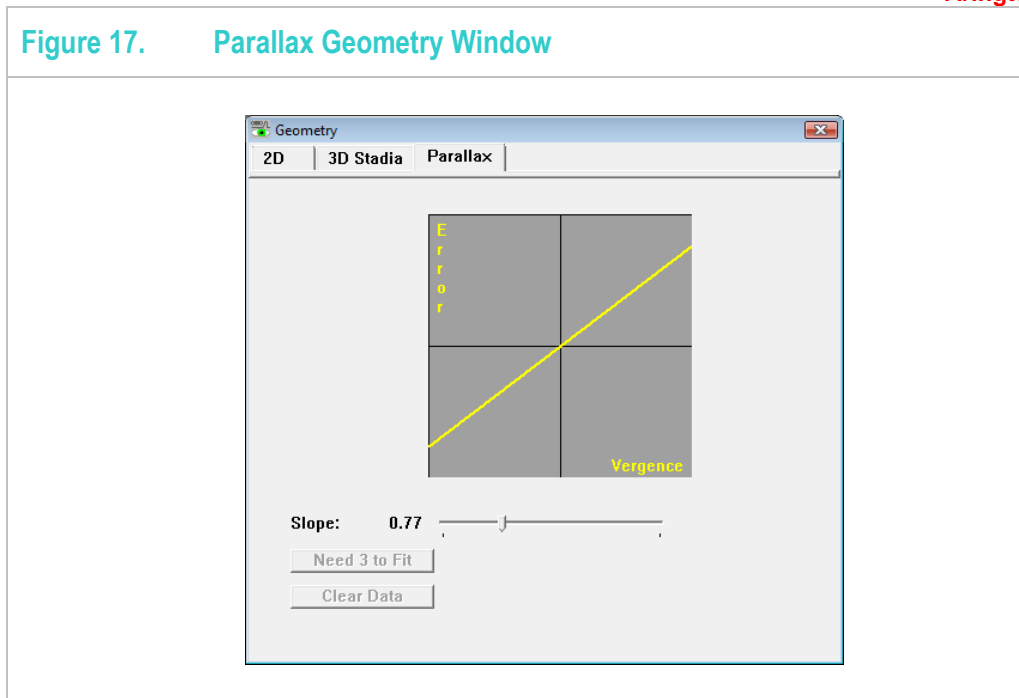
If you wish to fine tune this function, there are two ways to determine the appropriate parallax correction: (a) manual experimentation, (b) collecting data and fitting a line to it.

7.8.1 Manual Adjustment

After calibrating in binocular mode to the SceneCamera, select the **Regions Tab** on the **Controls Window**. Then select radio button **Parallax Correction** (in the **GazeSpaceMouseAction window**) which will bring up the **Parallax Geometry** window as in Figure 17 below.

Use the Slope slider to adjust the magnitude of the Parallax Correction function, as the subject looks near and far, until a setting is found that substantially eliminates the error between the supposed gaze point and the calculated gaze point.

Figure 17. Parallax Geometry Window



7.8.2 With Data

You can collect data to help you visualize the parallax error at different vergence angles. These (error,angle) pairs are graphically displayed in the 2D plot, as shown in Figure 18. You can also fit a line to this data.

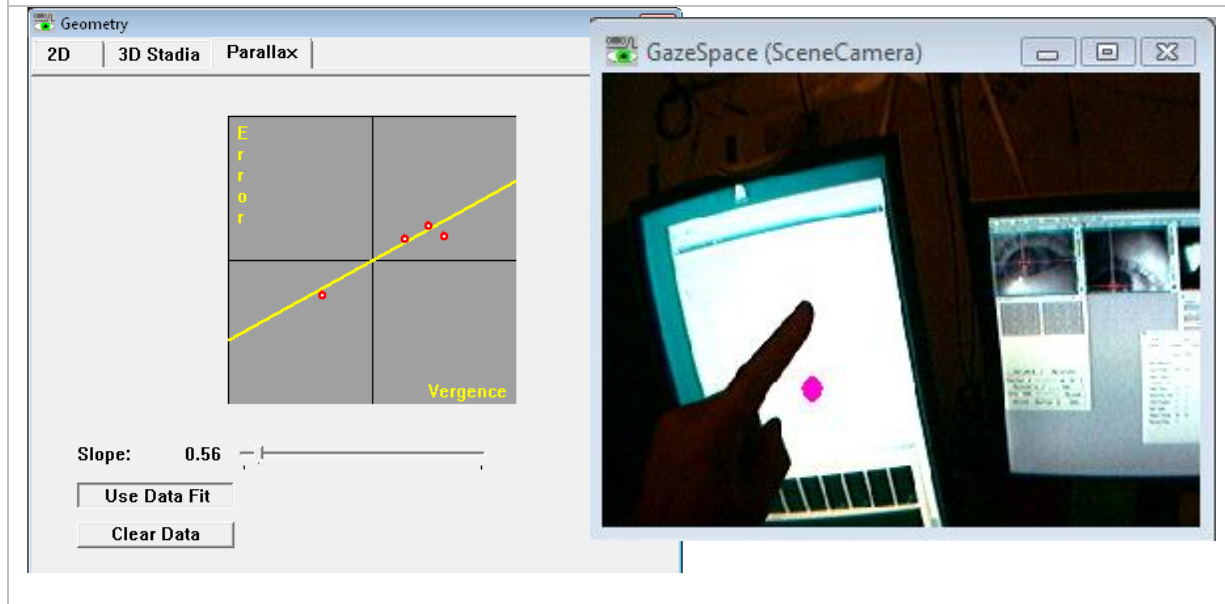
After calibrating in binocular mode to the SceneCamera, select the **Regions Tab** on the **Controls Window**. Then select radio button **Parallax Correction** (in the **GazeSpaceMouseAction** window well) which will bring up the **Parallax Geometry** window as in Figure 18 below.

1. Ask the subject to fixate on a spot at a close distance to the eye. e.g. 10cm
2. Then use the mouse in the GazeSpace window to click on the point where the subject is presumed to be looking . For example, in the image in Figure 18 below, if the subject was ask to look at the tip of the finger, then we click on the tip of the finger. The software records the difference between the calculated position of gaze (POG) and the point clicked to get the error term, the current vergence angle is also recorded.
3. Repeat steps 1. and 2. for a total minimum of 3 different distances to the eye. E.g. 10cm, 20cm, 1m, 2m, 3m. A minimum of three points are required before you will be allowed to fit a line to the data; in fact you should usually obtain at least six or eight.
4. Press the **Use Data Fit** button to apply that degree of parallax correction.

The default slope applied is 0.82, if you need to change this default use the following CLI command:

`parallaxCorrection_Slope` *float in range 0.5 – 1.5*

Figure 18. Parallax Geometry Window and Error Plotting



7.9 Dominant Eye

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

7.10 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 CLI saves these eye images in a folder named "Calibration":

```
SaveCalibrationEyeImages Yes
```

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

7.11 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.

7.11.1 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 CLI s the user can specify the presentation order of the calibration stimulus rectangles. See section 17.9.14 to 17.9.16.

7.11.2 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.

7.11.3 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 16EyeSpace 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 17.9.26 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.

7.12 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 17.9.22.

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 CLI s. See section 17.9.24.



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

7.13 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 19](#)) 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.

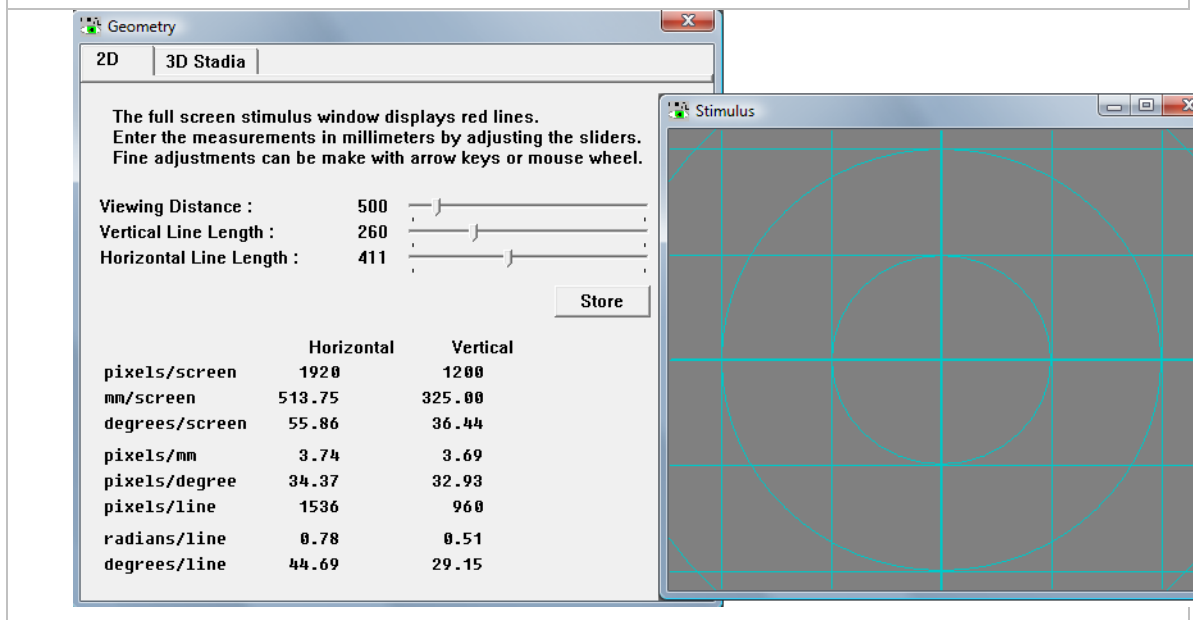
The units of measurement are arbitrary, but they must be consistent, e.g. all measurements in millimeters, or all measurements in inches. The [Apparatus Geometry](#) window also displays various numerical calculations that may be useful.

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 [Display](#) tab.

Figure 19. Apparatus Geometry Window



You can specify the minor and major axis separation displayed using the CLI `gridSpacing floatMinor floatMajor`

e.g.: `gridSpacing 2.5 10 // minor axes every 2.5 degrees, major axes every 10 degrees`

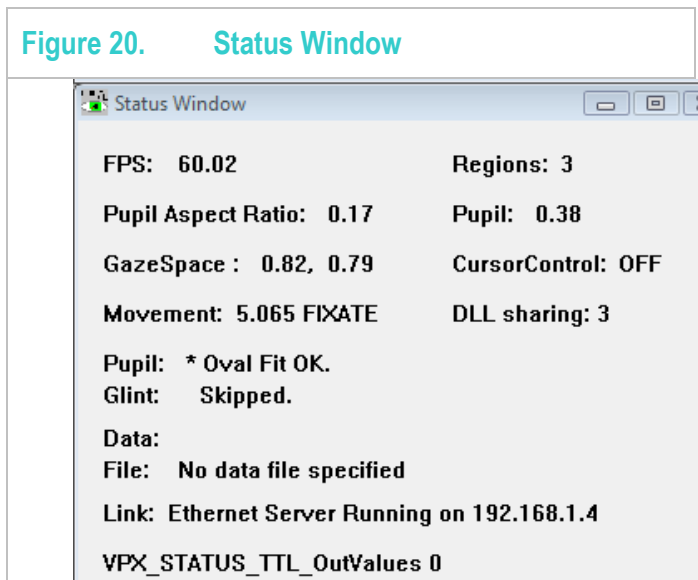
Chapter 8. Cursor Control

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 20. Status Window



Chapter 9. Ocular Torsion

9.1 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 21 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 22 below.

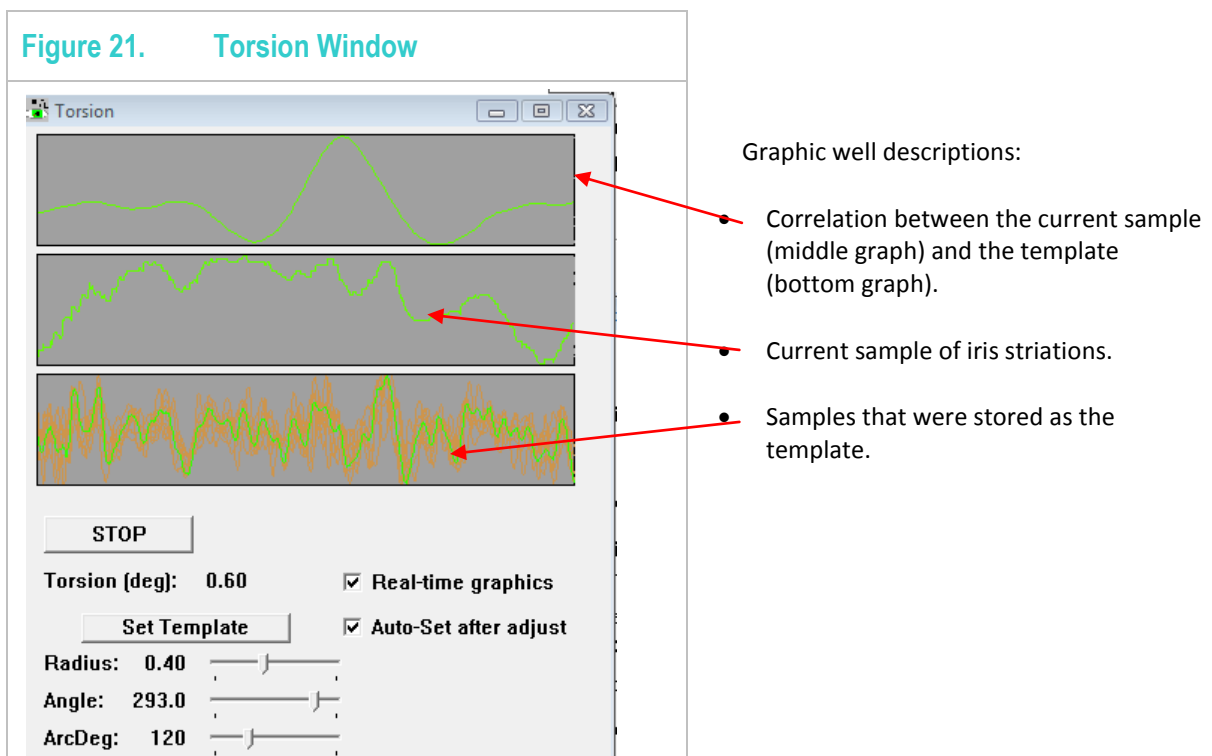
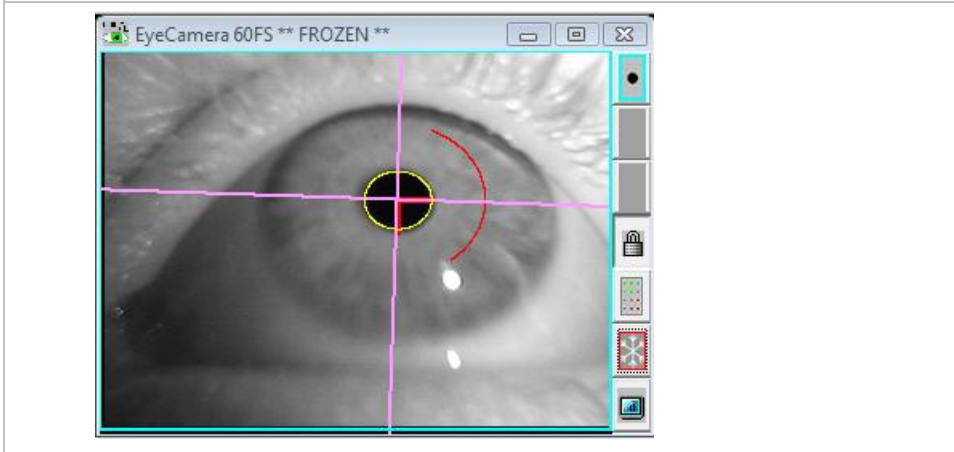


Figure 22. EyeCamera window with Torsion ON



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.

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

9.2 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 6.

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.

9.3 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 17.18 describes the torsion commands in detail.

9.4 **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 10. 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.

10.1 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 CLI , see section 17.23.2.

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.

10.2 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 17.4 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

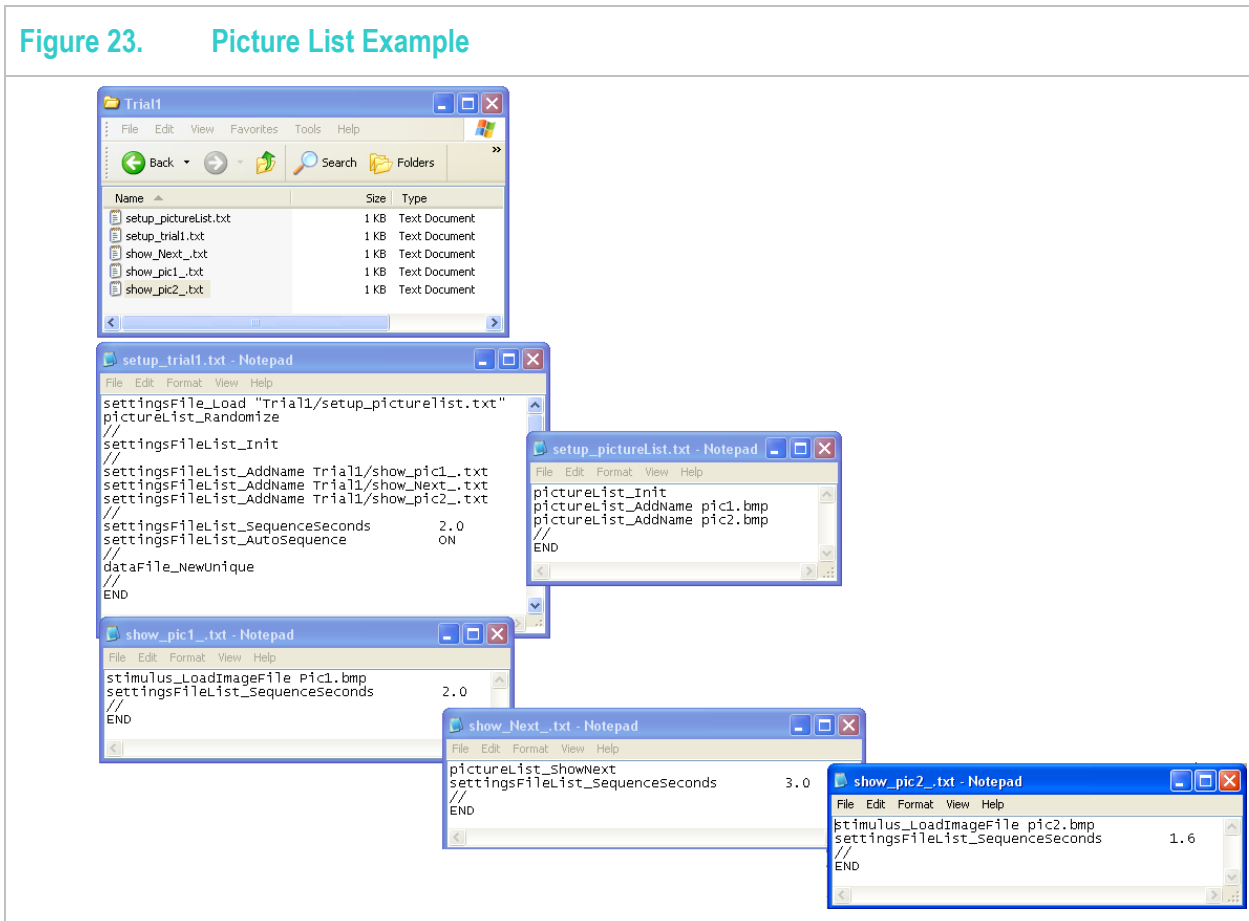
File > Image > Picture List > Restart PictureList Returns to the top of the currently loaded settings file



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 23 below demonstrates how settings files can be used very simply to present a series of stimulus images at intervals determined by the user.

Figure 23. Picture List Example



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

10.3 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 **Display** 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.

10.4 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 7.11.3, (unless custom calibration stimulus points are enabled, see sections 7.12) 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.
- ⦿ [Record Time](#) to display the record time when in SceneCamera ViewSource mode.

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.

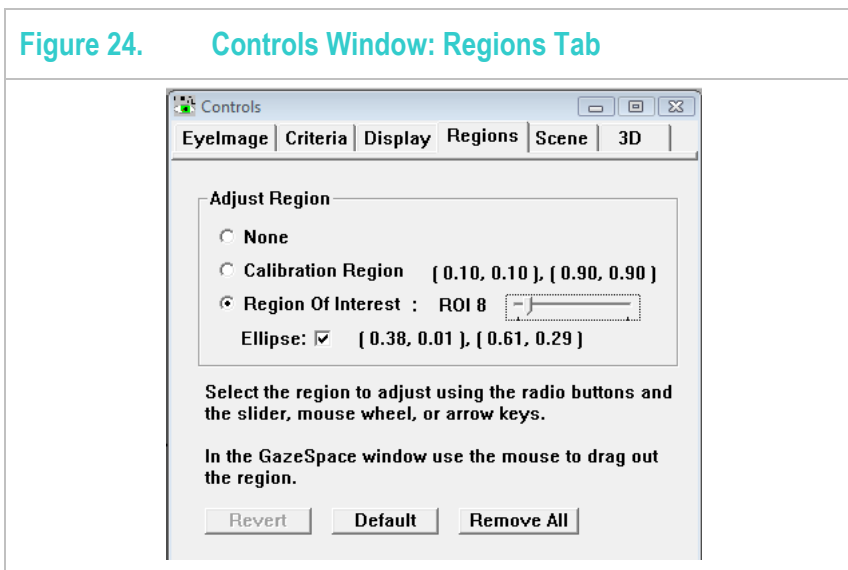
10.5 Regions of Interest (ROI)

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](#).

Figure 24. 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.

Use menu item **Remove All** to remove all the ROI set.

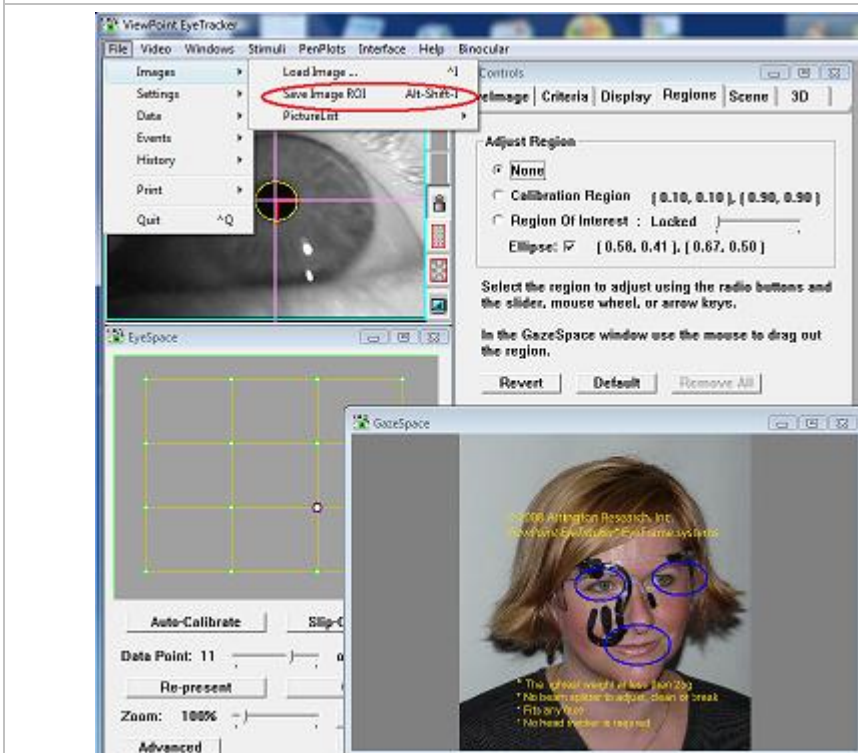
10.6 Associating an Image with Specific ROI

You can associate an image with the ROI for that image. Every time an image is loaded, e.g. ~/Images/MyImage01.mbp, ViewPoint also looked for the file ~/Images/MyImage01.bmp_ROI.txt that contains ROI specifications. See menu item: **File > Image > Save Image ROI**

For example, you have a series of stimulus images of faces with ROI that include the eyes and mouth of each face. You want to present the stimulus images in turn and examine how long the subject spends looking in each ROI for each face.

1. Set up your ROI for each image and select menu item: **File > Image > Save Image ROI**

Figure 25. ROI Image Association



2. Setup Picture Lists as described in 10.2 above. Then, when the images are loaded into the stimulus window and presented to the subject, the associated ROI are also loaded. The data file will then show if the position of gaze is in any of the ROI.

10.7 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 12.1.1.

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.

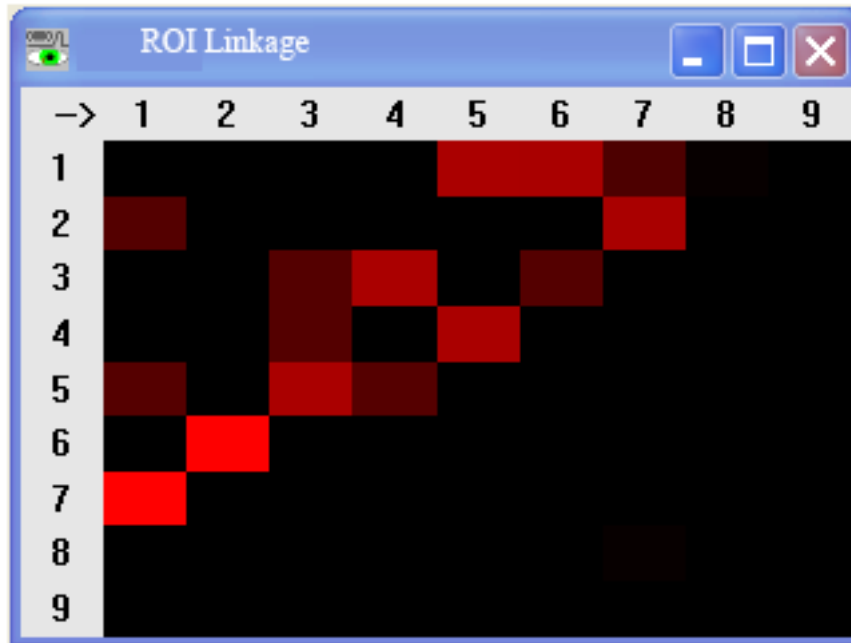
By default, smoothing does **not** influence the data values that are stored to file, because the real-time smoothing uses a trailing average technique, whereas post hoc data analysis should use a symmetrical smoothing technique. The user can choose to store the smoothed data by selecting menu item [File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

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

10.8 ROI Transition Statistics or Linkages

Transitions between different ROI are called **Linkages**. The linkage data can be viewed in a variety of ways. A graphical heat-map clearly shows *hot-spots* in the ROI transition. The heat map is displayed in the *ROI Linkage* window and is interesting to watch as it is updated in real-time. The ROI transition table is first normalized, then values are painted as intensities of red.

Figure 26. ROI Linkage Heat Map



The linkages can also be displayed in tabular form in the *History* window, which can be saved to file, as shown in the figure below. Use menu item: `Window > Dump Info > ROI Linkage Stats`.



Figure 27. ROI Linkages Data

Monday, September 11, 2006, 5:21:24 PM

ROI Linkage:

	1/	2/	3/	4/	5/	6/	7/	8/	9/
1]	3	4	1	2	1	2	2	1	0
2]	1	0	4	1	0	0	1	0	0
3]	0	1	0	5	0	0	0	0	0
4]	1	0	1	1	8	0	0	0	0
5]	0	0	0	1	0	9	1	0	0
6]	1	2	0	2	1	0	7	1	0
7]	1	0	0	0	1	3	1	8	0
8]	8	0	0	0	0	0	2	0	0
9]	0	0	0	0	0	0	0	0	0

ROI#	Count	%Hits	SumDur	MeanDur
1	15	0.17	1.34	0.09
2	7	0.08	1.09	0.16
3	6	0.07	0.74	0.12
4	12	0.13	1.43	0.12
5	11	0.12	1.16	0.11
6	14	0.16	1.52	0.11
7	14	0.16	1.45	0.10
8	10	0.11	0.75	0.08
9	0	0.00	0.00	0.00

Total: 89 1.00 9.48 0.11 <--- Overall

Careful inspection of the table will show that (a) the sequence must have started in ROI#1, because there are 16 transitions from ROI#1, but only 15 into ROI#1 and that (b) the sequence must have ended in ROI#5, because there are 12 transitions into it, but only 11 from it.

The Count values in the summary data is the sum of the TO columns, rather than the sum of the From rows.

10.9 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 Interface (CLI) in the *ViewPoint* application.

The same CLI 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 18, Settings Files in Chapter 13, and the Serial Port Interface in Chapter 16. Chapter 15 lists all of the available ViewPoint features, the GUI control, CLI and any applicable SD function. There are many features available through CLI s or SDK functions only.

10.10 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 11. Data Collection

11.1 Sampling Rate – Frame Grabber Modes PC-60

The *ViewPoint EyeTracker*® includes four 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. You may achieve better performance if you calibrate using the [High Precision Mode](#).

- | | |
|---|----------------------|
| <ul style="list-style-type: none"> ● Setup Mode:
Temporal Resolution: 30Hz
Internal Processing: 320 x 240 | Interlaced |
| <ul style="list-style-type: none"> ● High Precision Mode:
Temporal Resolution: 30Hz
Internal Processing: 640 x 480 | Interlaced |
| <ul style="list-style-type: none"> ● High Speed Wide Mode:
Temporal Resolution: 60Hz
Internal Processing: 640 x 240 | De-interlaced |
| <ul style="list-style-type: none"> ● High Speed Normal Mode:
Temporal Resolution: 60Hz
Internal Processing: 320 x 240 | De-interlaced |

The monitor icon > Mode at the bottom of the [EyeCamera](#) window tool bar can be used to select the required operating mode. 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.

The [EyeCamera](#) window EyeImage is always displayed a 320 X 240, regardless of the mode selected.

11.2 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 17.23.3.

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 ...](#)(Ctrl-U) 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 CLI : [dataFile_NewUniqueExtension](#) described in section 17.2.2. The file name created will include the date and time started starting with the year.

[File > Data > Pause Data Capture](#) (Ctrl-P) 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 pressing the freeze button on the *EyeCamera* window which stops all processing and recording of data to file. [File > Data > Close Data File](#) (Ctrl-W) terminates the writing of data and closes the open file.

The command line interface for data files is described in section: 17.2.

11.3 Data File Format

11.3.1 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.

11.3.2 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 11.3.4.

11.3.3 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_asynchStringData YES
dataFile_InsertString      "CAT_A (asynchronous)"
dataFile_InsertString      "CAT_B (asynchronous)"
dataFile_asynchStringData NO
dataFile_InsertString      "DOG (synchronous)"
```

produce the following data file lines:

Table 6. DataFile with Synchronously and Asynchronously Inserted Markers

10	2.5322	33.3327	0.4924	0.4746	0	0.1469	0.8936	1	0.6339	74	
10	2.5656	33.406	0.4924	0.4746	0	0.1469	0.8936	1	0.6673	75	
12	2.577245	CAT_A (asynchronous)									
12	2.580258	CAT_B (asynchronous)									
10	2.6419	76.3175	0.4924	0.4746	0	0.1469	0.8936	1	0.7437	76	DOG (synchronous)
10	2.6513	9.3698	0.4924	0.4746	0	0.1469	0.8936	1	0.753	77	
10	2.6657	14.4061	0.4867	0.4734	0	0.15	0.875	1	0.7674	78	

11.3.4 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. The primary record types are as follows:

- ... **Tag #10:** EyeData containing a variable number of columns depending on the options selected for data collection. Table 7
- ... **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 8
- ... **Tag # 12:** An ASCII character string asynchronously inserted during certain events. For example, by the CLI in response to certain commands.

E.g. “pictureList_ShowNext.”, “dataFile_InsertString picture of a cat” from a settings file, or an external program. Table 11



- ... **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 **Error! Reference source not found.**
- ... **Tag # 6:** A three character data column identifier generated by *ViewPoint*. Refer to **Error! eference source not found.**
- ... **Tag # 14:** Asynchronously inserted head tracker data.
- ... **Tag #16:** Picture Image File Name.
- ... **Tag # 777:** Movie Frame Number.

Note: The user can insert data from their own sources into a ViewPoint data file with their own specified tag number in the range 800-899. Use the CLI DataFile_InsertUserTag. The insertion is done asynchronously with respect to the eye movement data records and the insertions are uniquely time stamped.

Table 7. 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 11.8, 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 printable ASCII character, e.g., {a-z, A-Z,0-9,=,#,+,%, etc.}. See Table 8, below.
pXraw		Only displayed if the option to save raw data is checked.
pYraw		Only displayed if the option to save raw data is checked.
gXraw		Only displayed if the option to save raw data is checked.
gYraw		Only displayed if the option to save raw data is checked.

Table 8. Asynchronous Marker Record Structure, Tag = 2

<i>Column #</i>	<i>Type</i>	<i>Value</i>
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 9. Asynchronous HeadTracker Data Record Structure, Tag = 14

<i>Column #</i>	<i>Type</i>	<i>Value</i>
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 10. String Data Record Structure, Tag = 3

<i>Column #</i>	<i>Type</i>	<i>Value</i>
1	integer	3 = integer indicates data structure type 3
2	string	File header information, an ASCII character string generated by <i>ViewPoint</i>

Table 11. String Data Record Structure, Tag = 12

<i>Column #</i>	<i>Type</i>	<i>Value</i>
1	integer	12 = integer indicates data structure type 12
2	float	Time stamp
3	string	ASCII character string from "dataFile_InsertString", etc.



Table 12. Column Header Data Record Structure (Tag # 5 and Tag # 6)

Column #	Type	Value		
1	Integer	5	6	Eye
2 – n	String	Total Time	ATT	<div style="display: flex; flex-direction: column; align-items: center;"> <div style="margin-bottom: 10px;"> } Eye_A </div> <div style="margin-bottom: 10px;"> } Eye_B </div> <div style="margin-bottom: 10px;"> } Head Tracker Option only </div> <div style="margin-bottom: 10px;"> } </div> </div>
		Delta Time	ADT	
		X-Gaze	ALX	
		Y-Gaze	ALY	
		ROI	ARI	
		Pupil Width	APW	
		Pupil Aspect	APA	
		Quality	AQU	
		Fixation	AFX	
		Total Time	BTT	
		Delta Time	BDT	
		X-Gaze	BLX	
		Y-Gaze	BLY	
		ROI	BRI	
		Pupil Width	BPW	
		Pupil Aspect	BPA	
		Quality	BQU	
		Fixation	BFX	
		X	HPX	
		Y	HPY	
		Z	HPZ	
		Yaw	HAY	
		Pitch	HAX	
		Roll	HAZ	
		Record Count	CNT	
		Marker	MRK	

11.4 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 Chapter 8 Cursor Control Feature)

11.5 Raw Data

Raw (unmapped EyeSpace) data can be saved to the data file. Select menu **File > Data > Include Raw (unmapped EyeSpace data)**.

11.6 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 eye data stream entry. This represents the time that the software has finished processing each frame and the eye movement data is available.

Note that the crystal clock in the camera is very precise, so the field and frame delivery from the camera is very regular at 16.6667 msec and 33.3333 msec respectively. Variability of the time stamps is due to CPU load handling.

11.7 Region of Interest (ROI)

Often one is only interested in whether or not the direction-of-gaze is within a specific region, or more generally, which of several possible regions. Regions of Interest (ROI), sometimes called Areas of Interest (AOI), may be defined as described in 10.5 . 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 be overlapping or nested, 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.

Note: New projects should not use ROI #0. While this is adequate as an ROI condition, this cannot be signed as +/- to indicate ROI into and out of events respectively.

11.8 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:

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

Refer to 17.25.5

11.9 Pupil Diameter













The value of PupilWidth is width of the ellipse that is fit to the pupil, normalized with respect to the width of the EyeCamera window. Using normalized values allows the user to switch between video modes (e.g. 320x240, or 640x480) without affecting the data. Consequently, if the pupil ellipse horizontally spans the full width of the window, the width value would be 1.0.

The Ellipse method fits a rotated ellipse to the pupil; with this method the PupilWidth is the length of the major axis (the longest axis). Note that the minor axis will become small as the eyeball rotates away from the optical axis of the camera, as a cosine function of rotation, becoming zero at 90 degrees. The major axis does not suffer from this problem. Note also that the rotated ellipse allows a diagonal fit to the full size of the EyeCamera window, which allows the PupilWidth value to exceed 1.0.

The older OvalFit method fits an unrotated ellipse to the pupil; with this method the PupilWidth is the horizontal width of the pupil.

Obviously the actual pupil diameter will depend on the camera placement relative to the eye. Figure 28 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.

Figure 28. Artificial Pupil Diameters

 1/8"	 3/16"	 1/4"	 5/16"	 3/8"		
 2 mm	 3 mm	 4 mm	 5 mm	 6 mm	 7 mm	 8 mm

11.10 Pupil Aspect

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

11.11 Display Screen Geometry

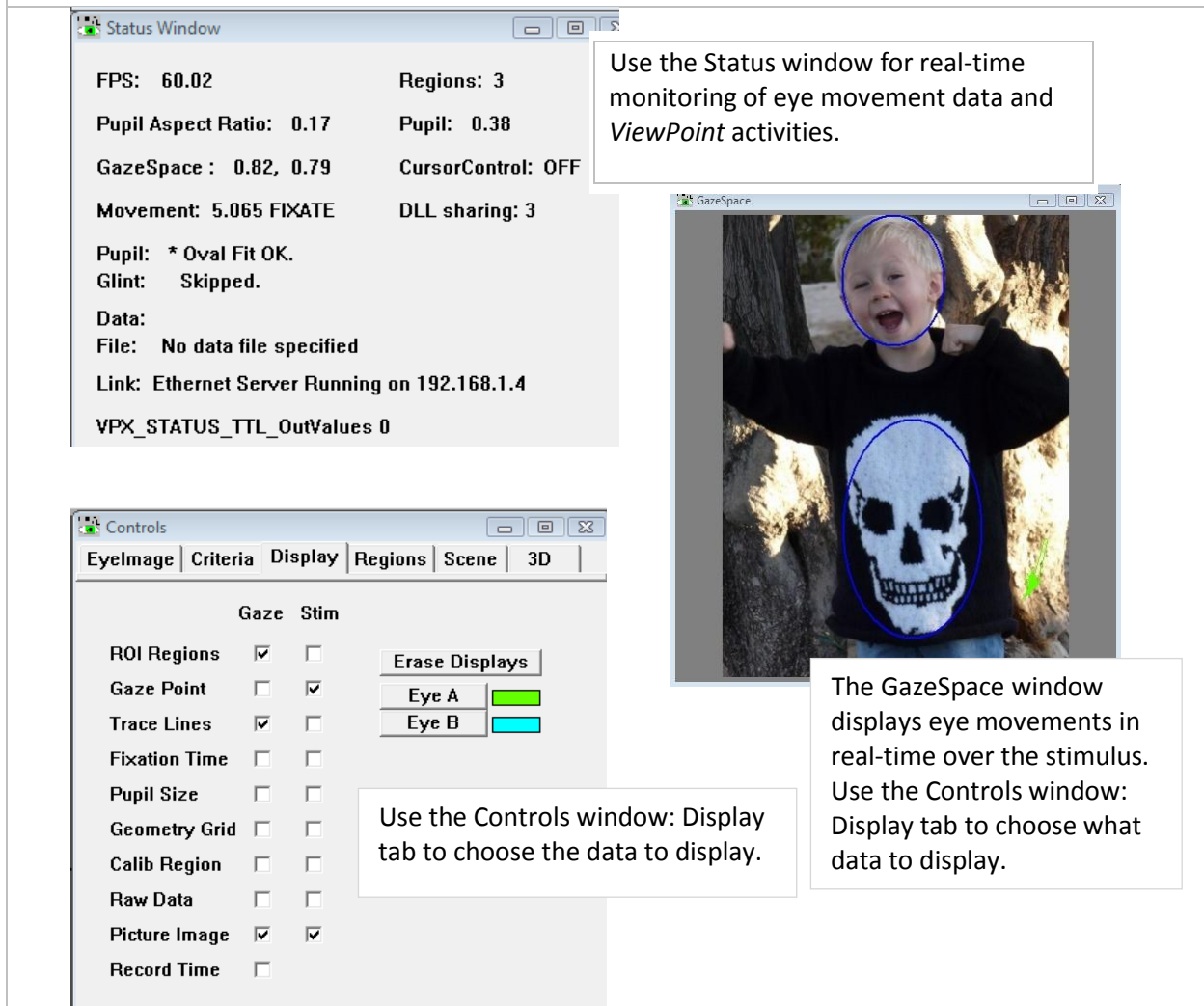
The data file also contains header information that specifies the results of the [GeometryGrid](#) calculations for screen size and viewing distance. The screen size values saved will be the values calculated for the entire screen width and height, not the measured lengths of the lines (that are 20%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

Chapter 12. Data Analysis

12.1 Real-Time

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

Figure 29. Status and GazeSpace Windows



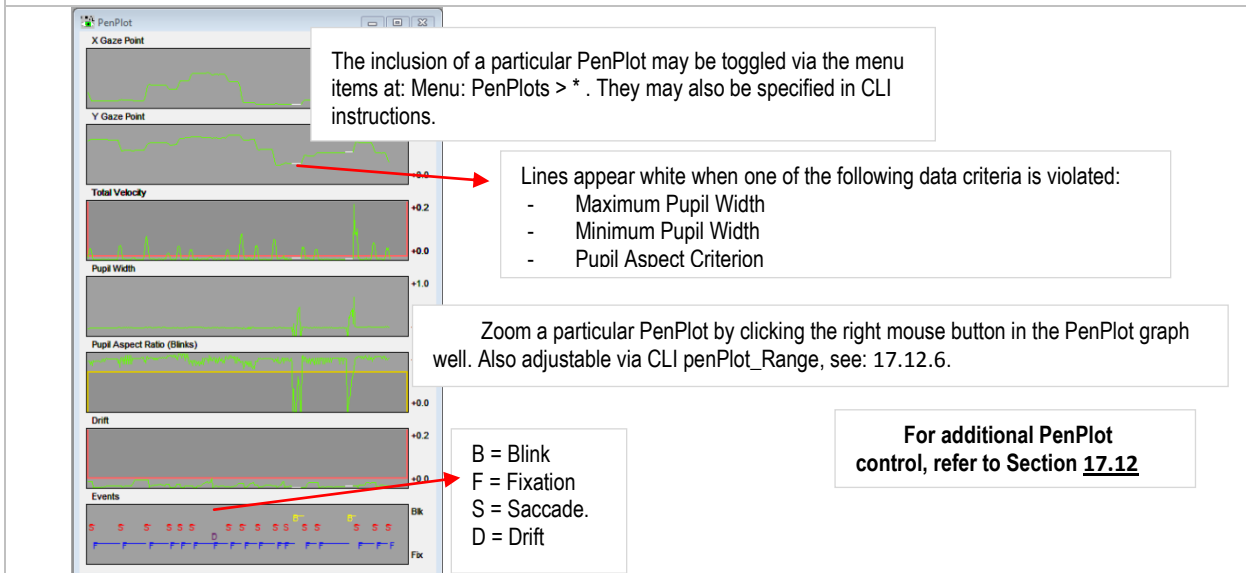
The figure displays three windows from the ViewPoint EyeTracker software:

- Status Window:** Shows real-time monitoring data including FPS (60.02), Pupil Aspect Ratio (0.17), GazeSpace coordinates (0.82, 0.79), Movement (5.065 FIXATE), Pupil (0.38), and CursorControl (OFF). It also indicates that the data file is not specified and the system is running on an Ethernet server at 192.168.1.4.
- GazeSpace Window:** Displays a video feed of a child in a skull-patterned shirt. Blue circles highlight the child's eyes and the skull graphic, showing real-time eye movement tracking over the stimulus.
- Controls Window:** Features a 'Display' tab with various data options. The 'Display' tab is selected, and the 'Picture Image' option is checked under both 'Gaze' and 'Stim' columns. Other options like ROI Regions, Gaze Point, Trace Lines, and Fixation Time are also visible.

Callout boxes provide additional context:

- Use the Status window for real-time monitoring of eye movement data and *ViewPoint* activities.
- Use the GazeSpace window displays eye movements in real-time over the stimulus. Use the Controls window: Display tab to choose what data to display.
- Use the Controls window: Display tab to choose the data to display.

Figure 30. PenPlot Window



12.1.1 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 [Criteria](#) tab or with CLI s. 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 EMA method uses the following algorithm:

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

NOTES:

1. SMA is the default setting.
2. The data is stored UNSMOOTHED unless specified by the user.
3. The successfully set method is currently reported in the Event History window and the Status Window.

12.2 Fixation, Saccade, Drift and Blinks

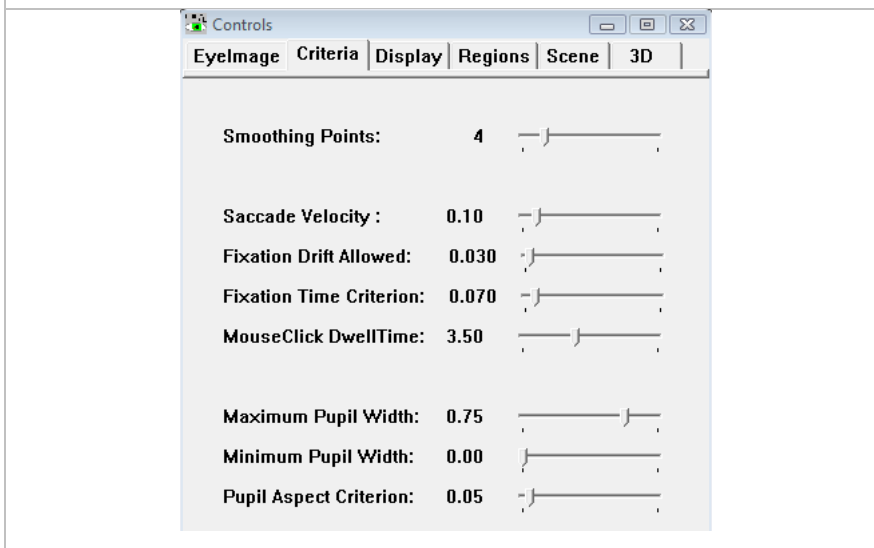
12.2.1 Velocity Threshold

The Saccade Velocity slider on the [Controls](#) window [Criteria](#) tab (See Figure 31) 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.

Figure 31. Controls Window: Criteria Tab



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.

12.2.2 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.

12.2.3 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 : Criteria tab : Drift Allowed *slider*

CLI: 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

CLI: penPlot +DRIFT

12.2.4 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.

12.2.5 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

12.2.6 SDK

The real time values for all data points are available via the DLL based SDK. See section 18.10.

12.3 Post-Hoc

The data file format is described in Chapter 11.

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](#)

CLI: `dataFile_CloseAndAnalyze`

12.4 Summary Data

Summary data can be viewed using menu item: [Window > Dump Info > ROI Linkage Stats](#). This includes fixation and blink summary data as well as ROI transitions. Refer to [Figure 27](#)

Chapter 13. 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 Interface (CLI). 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 CLI 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 CLI 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.*

13.1 CLI 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.

13.2 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 CLI to be displayed in the [Event History](#) window.

[File > Settings > Save Window Layout](#)
saves the size, location and z-ordering of all ViewPoint windows.

13.3 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.

13.4 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`

13.5 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 CLI (`settingsFileList_Init`, etc.) described in section SettingsFileList commands 17.17. Start and control the sequencing via the menu item [File > Settings File > SettingsFileList](#). *Hint*. It is useful to assign FKey commands for these.

13.6 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
```

13.7 CLI s

The same Command Line Interface (CLI) 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 13 for details about using CLI s.

13.8 Associating CLI s with FKeys

CLI s can be associated with FKeys. These associations can be viewed in the Info panel: menu Help > Info > **ShortCuts** tab. Refer to 17.24.1

13.9 Simple Command Line Interface Program

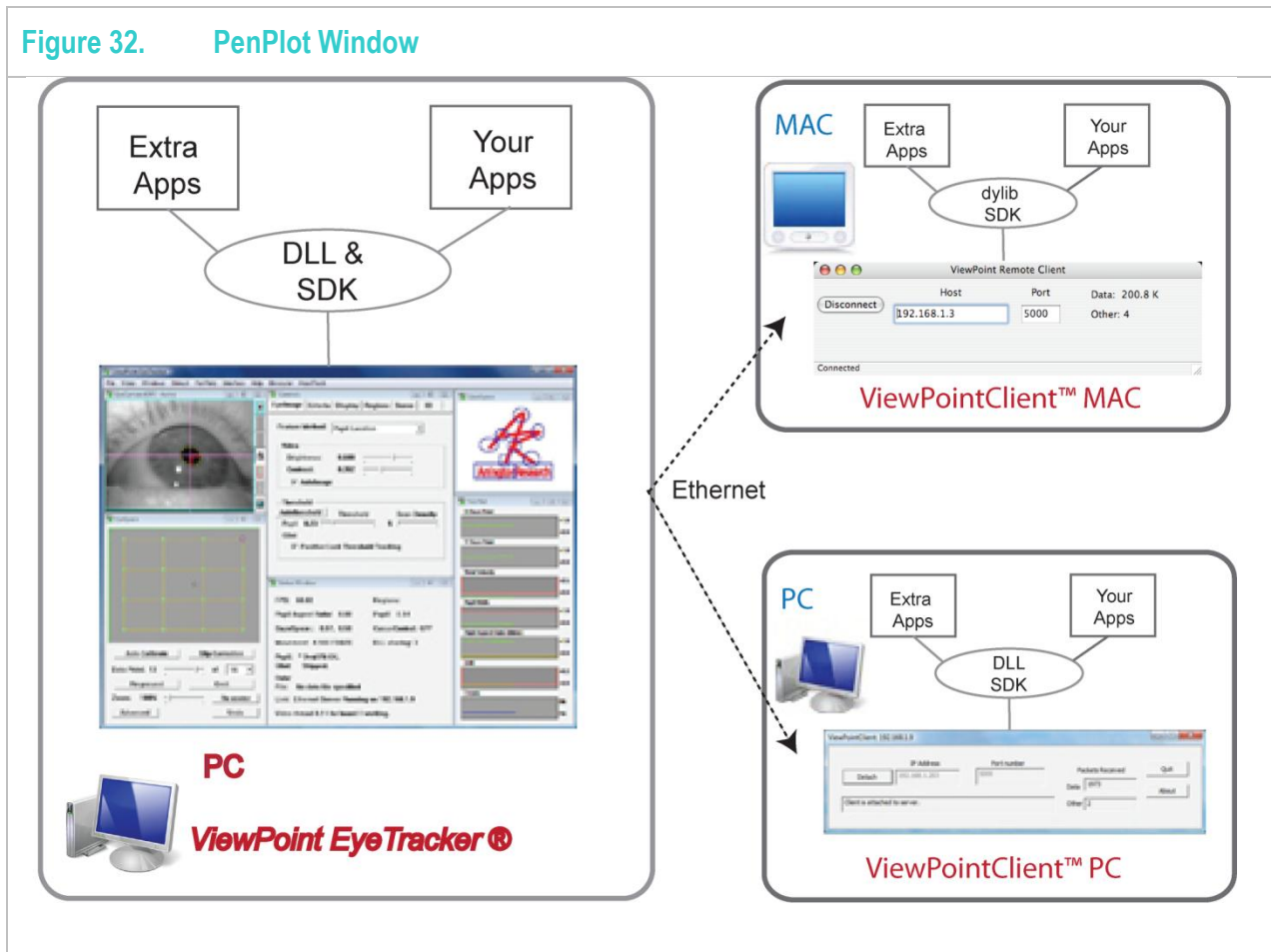
Select menu Windows > Command Line Interface to open a simple interface to send CLI s.

Chapter 14. Ethernet Communication between PCs

The *ViewPoint EyeTracker* communicates with other “layered” applications running on the same computer, via a dynamically linked library (dll). The dll is part of the *ViewPoint* Software Developers Kit (SDK) that contains high level functions that allow the user to seamlessly and easily interface their programs with the eye tracker. *ViewPointClient* is a program that runs on a remote computer and communicates with the *ViewPoint EyeTracker*. *ViewPointClient* interfaces to a copy of the dll and exchanges data just like the *ViewPoint EyeTracker* does, but it typically takes less than one percent of cpu resources. This means that the same “layered” applications can be used on a remote computer just as easily as on the same computer. The *ViewPoint EyeTracker* includes an ethernet server; the *ViewPointClient* establishes an Ethernet link with this server.

ViewPointClient is designed to replace the older *RemoteLink* program that provided similar functionality over a serial port (COM port) interface. The new *ViewPointClient* and Ethernet interface provide more extensive data synchronization as well as faster and more reliable data delivery.

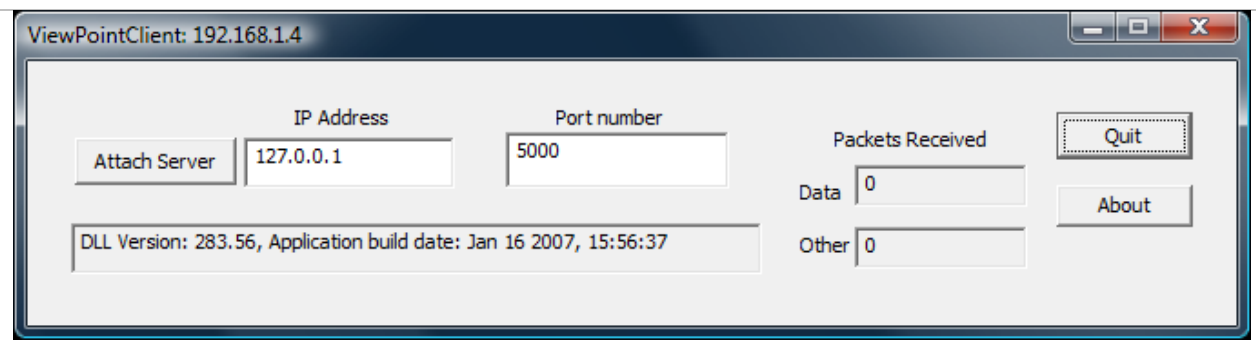
Figure 32. PenPlot Window



14.1 How to use ViewPointClient

1. Copy the *ViewPointClient.exe* application and the *VPX_InterApp.dll* into the same folder on a second computer.
2. Start the *ViewPoint EyeTracker* application. This automatically starts the built in server.
3. Open the *ViewPointClient.exe* application; the following window will be displayed.

Figure 33. ViewPointClient Window

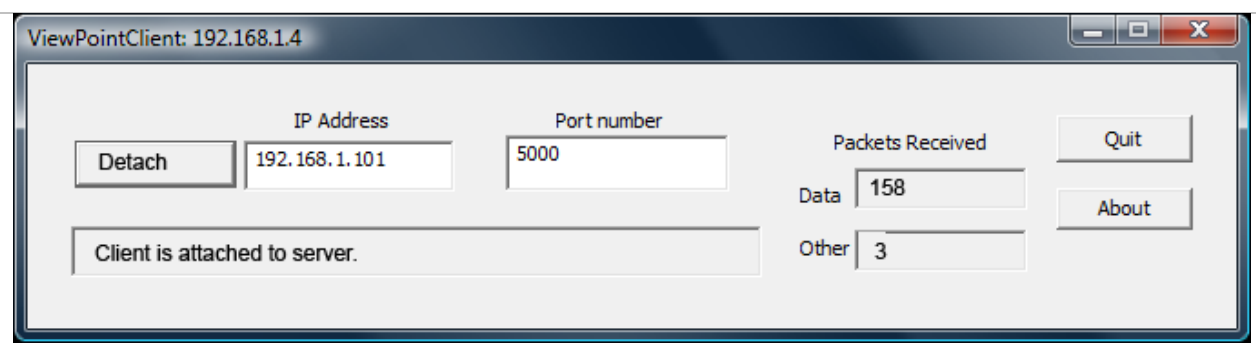


4. Enter the IP Address (Eg: 192.168.1.101) of the computer on which *ViewPoint* is running and click "Attach Server". It is rarely necessary to change the Port number, so start by leaving it as the default value. Note: 127.0.0.1 address can be used to "self-loop", but should rarely be used. (Refer to **LoopBack** Section). There is no name resolution, you need to use the IP address not the computer name.

Note: to determine the IP address of the computer on which ViewPoint is running, go to the Windows control panel and select network connections. You will find all of the required information under the properties of the local area network connection.

5. The *ViewPointClient* application should look like the following.

Figure 34. ViewPointClient Window when attached



The **Data** field indicates the number of eye tracking data packets received. If the **Data** field is 0, or remains constant, then probably either *ViewPoint* is in **Freeze** video mode, or the *ViewPoint* menu item: *Interface > Send Data > ** options needs to be set to **Streaming Data**. This field will continuously increment when it is receiving eye tracking data in real-time.

14.2 Third Party Applications

Copy the **Interfaces** folder to the folder containing *ViewPointClient* and *VPX_InterApp.dll* on the second machine. To run any of the third party applications, please consult their respective documentations.

14.3 Layered Applications

The executable files for the demo-applications are in the *ViewPoint/ExtraApps/** folder. You can copy these to the second machine into the folder containing *ViewPointClient* and *VPX_InterApp.dll*. Run the applications just as you would if they were in the folder where *ViewPoint* is running. These include:

- VPX_SimpleC.exe
- VPX_Win32_Demo.exe
- VPX_MFC_Demo.exe
- VPX_Basic_Demo.exe

You can also write your own layered applications to run on on either machine. We recommend that you start with the sample code and projects in the *ViewPoint/SDK/** folder.

14.4 Changing the Port Number

The current default Ethernet port is 5000 and should not need to be changed. However if it is changed, the server will be stopped and all client connections will be lost, then the server will be restarted; any clients will need to be attached again manually. This is done with the following command line instructions:

```
ethernet_setPortNumber unsignedInteger
```

Note: Go to the dos command prompt into c route directory. Use command netstat to access network connection information. Netstat – h lists all available commands.

14.5 Running the Server

It does not matter whether the *ViewPoint* or the *ViewPointClient* application is launched first, however the *ViewPoint* application and its built-in server must be running before the *ViewPointClient* application can attach to the server. Normally the server can remain running and should not need to be turned off, however if this seems necessary, it is accomplished with the following command line instruction:

ethernet_server boolValue

14.6 Loopback

The ViewPoint EyeTracker server and ViewPointClient applications can communicate with one another on the same machine using the Ethernet loopback interface available at address 127.0.0.1. The two applications must be in separate folders, each with their own VPX_InterApp.dll. This can be useful for testing, however it is not as efficient as using a single copy of the VPX_InterApp.dll for interprocess communication on the same machine.

14.7 Firewalls

Many computers now run a firewall to help prevent unauthorized access. You may see a dialog box message similar to the one below if the computer running ViewPoint is protected by a firewall program. You must unblock the program, i.e., press the Unblock button, for the ViewPoint Ethernet server to communicate with the ViewPointClient programs. You may also need to disable any virus checking software.



14.8 Hub, Switch, Router, or a Crossover cable ?

If you already have a Hub, Switch, or Router, then any of these will work fine. If you plan to purchase a new one, then we suggest that you choose a Switch or Router as these are traditionally more efficient. Any of these will allow several computers running ViewPointClient to the ViewPoint EyeTracker built-in server all at the same time. If you only want to connect two computers then you can simply use an Ethernet Crossover cable, which does not require any kind of Hub, Switch, or Router, but simply plugs into the Ethernet port of each computer.

Chapter 15. Ethernet Communication: PC- Mac

The *ViewPoint EyeTracker* communicates with other “layered” applications running on the same computer, via a dynamically linked library (dll). The dll is part of the *ViewPoint* Software Developers Kit (SDK) that contains high level functions that allow the user to seamlessly and easily interface their programs with the eye tracker.

ViewPointClient for MAC is a program that runs on a remote MAC computer and communicates with the *ViewPoint EyeTracker*. *ViewPointClient for MAC* interfaces to a dynamic library (dylib) and exchanges data just like the *ViewPoint EyeTracker* does. The *ViewPoint EyeTracker* includes an ethernet server; the *ViewPointClient* establishes an Ethernet link with this server.

15.1 Overview:

The *ViewPoint* distributor application does two main things: (a) updates the data in the shared memory of the library, (b) makes sure that VPX_SendCommand instructions are sent to the *ViewPoint* Command Line Interface (CLI). Examples of distributor applications are: the *ViewPoint EyeTracker*® itself, the new *ViewPoint Client* application, and the legacy *ViewPoint RemoteLink™* serial port application.

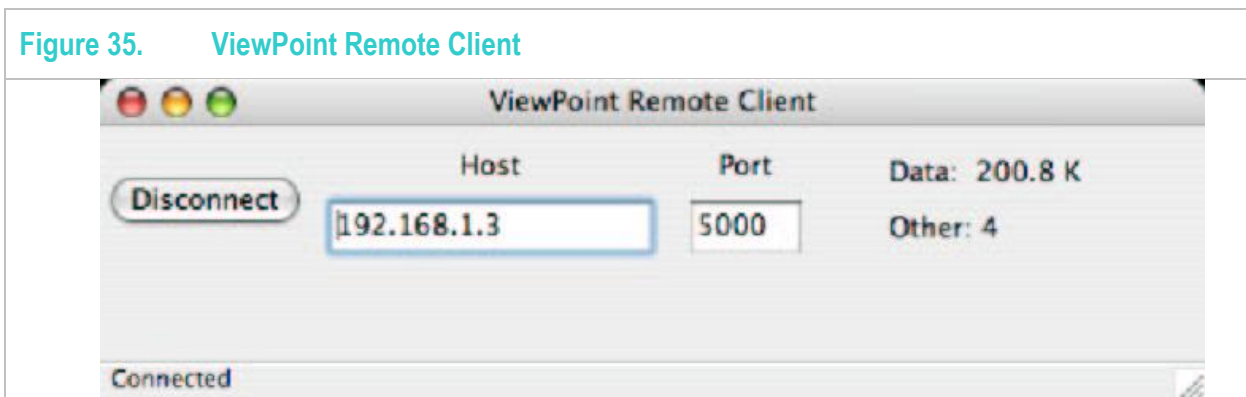
Only one *ViewPoint* distributor application should be attached to the library at a time. Conversely, several other layered applications, or application interfaces, may be connected simultaneously, for example: custom built applications that use the *ViewPoint* software developers kit (SDK), the *ViewPoint EyeTracker* Toolbox for MATLAB, etc.

Figure 35 shows the interface to the *ViewPoint Remote Client* distributor application. The user should set the Host IP address. < augment >

Data: indicates a count of the data packets that are usually continuously streaming

Other: indicates a count of non-data packets, that are sent occasionally, e.g. for a status change.

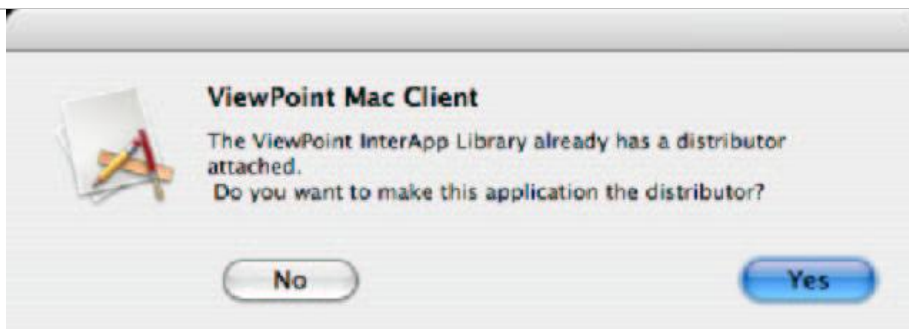
Figure 35. ViewPoint Remote Client





When a distributor application attaches, it sets a flag indicating this. If the distributor application terminates unexpectedly, the flag will not be reset properly and the message box shown in Figure 36 will appear. Press **Yes** to reset this flag.

Figure 36. Client Termination Message Box



Only one ViewPoint distributor application should be attached to the shared library at a time. You may see this message box if the distributor application terminates unexpectedly, press Yes to proceed.

15.2 How to use ViewPointClient for MAC:

15.2.1 METHOD 1

1. Open a terminal Window using the **Terminal** application in the Mac **Applications/Utilities/** folder.
2. Drag the text clipping document **export vpx library** from the **ViewPoint** folder to the terminal window, click the terminal window to set keyboard focus, and press carriage return to enter the command. The terminal window should look like **Error! Reference source not found..**

Figure 37. Specifying the library path

```
Terminal — bash — 77x7
Last login: Thu Feb 23 10:40:04 on ttty4
Welcome to Darwin!
ARIG4B:~ dev$ export DYLD_LIBRARY_PATH=/Users/dev/Desktop/ViewPoint/SDK/
ARIG4B:~ dev$
```

3. Make sure `export vpx library` is set for the proper path in the release.

OR

4. Must put the library **libvpx_interapp.dylib** into **/usr/local/lib/**

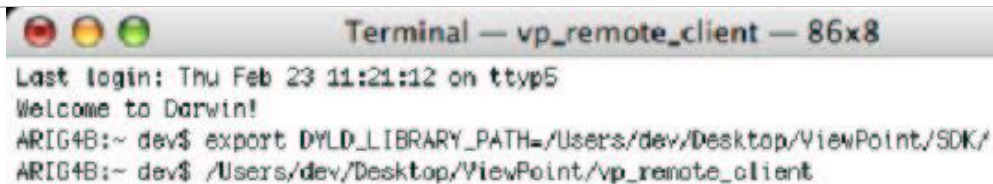
```
cd /usr/local/lib/
cp libvpx_interapp.dylib
```

and change the permissions as with

```
chmod 755 libvpx_interapp.dylib
```

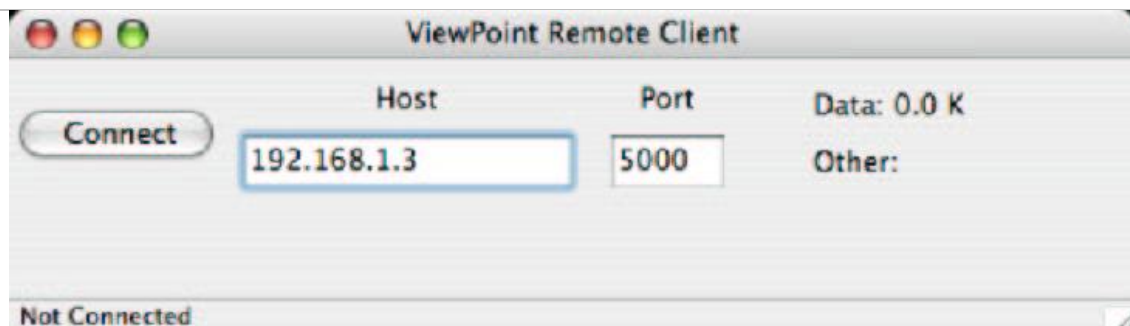
5. Drag the application **vp_remote_client** from the **ViewPoint** folder to the terminal window, click the terminal window to set keyboard focus, and press carriage return to enter the command. The terminal window should like Figure 38 and the window in Figure 39 should show up.

Figure 38. Launching the *ViewPointClient* application.



```
Terminal — vp_remote_client — 86x8
Last login: Thu Feb 23 11:21:12 on ttys5
Welcome to Darwin!
ARIG4B:~ dev$ export DYLD_LIBRARY_PATH=/Users/dev/Desktop/ViewPoint/SDK/
ARIG4B:~ dev$ /Users/dev/Desktop/ViewPoint/vp_remote_client
```

Figure 39. *ViewPointClient*



Notes:

The following error implies that the `libvpx_interapp.dylib` has not been exported.

```
dyld: Library not loaded: libvpx_interapp.dylib
```




Referenced from:/Users/dev/Desktop/ViewPoint/VPX_IntrApp/vp_remote_client

Reason: image not found

Trace/BPT trap

Make sure the library is exported as mentioned in Step 2.

6. Enter the IP Address (Eg: 192.168.1.3) of the computer on which *ViewPoint* is running and click "Connect". There is no name resolution, you need to use the IP address not the computer name.

NOTE: The Intel based MACs are not currently supported.

15.3 Third Party Applications

Copy the **Interfaces** folder to the folder containing *ViewPointClient* and *VPX_InterApp.dll* on the second machine. To run any of the third party applications, please consult their respective documentations.

15.4 Layered Applications

The executable files for the demo-applications are in the ***ViewPoint/ExtraApps/**** folder. You can copy these to the second machine into the folder containing *ViewPointClient* and *VPX_InterApp.dll*. Run the applications just as you would if they were in the folder where *ViewPoint* is running. These include:

- VPX_SimpleC.exe
- VPX_Win32_Demo.exe
- VPX_MFC_Demo.exe
- VPX_Basic_Demo.exe

You can also write your own layered applications to run on on either machine. We recommend that you start with the sample code and projects in the ***ViewPoint/SDK/**** folder.

Chapter 16. Serial Port Communication

If at all possible, use Ethernet communication as we may not continue to support serial communication. It is an old technology and many new computers do not ship with a serial port.

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

16.1 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.

16.2 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.
- drift

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

Use the CLI `EventFilterOptions` to control what constitutes an event.

eventFilterOptions +saccade +roi -drift +fixation // use + or - to turn on or off

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.

16.3 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.

16.4 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.

16.5 Serial Protocol

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

Setting	Value
Baud rate	56K (57,600)
Data bits	8
Parity bits	0 (no parity)
Stop bits	1 (one stop bit)

16.6 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:

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.

16.7 Serial Packet Data Structures

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

<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 Interface
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*

16.8 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 );
```

16.9 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;

```

16.10 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', 'O', 'O' };
    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
 CLI : verbose +serialReceive // +serialSend

16.11 RemoteLink™

The *ViewPoint EyeTracker* communicates with other “layered” applications running on the same computer, via a dll library. The dll library is part of the *ViewPoint* Software Developers Kit (SDK)



that contains high level functions that allow the use to seamlessly and easily interface their programs with the eye tracker.

RemoteLink is a program that runs on a remote computer connected by a serial connection and communicates with the *ViewPoint EyeTracker*. *RemoteLink* interfaces to the same dll library and exchanges data just like the *ViewPoint EyeTracker* does. This means that the same “layered” applications can be used on a remote computer just as easily as on the same computer.

16.11.1 RemoteLink Setup

Copy the *VPX_InterApp.dll* file from the *ViewPoint* folder into the folder where you will be running *RemoteLink* and your interfacing applications.

16.11.2 Data Transfer Formats

- Private Format

Newer versions of *RemoteLink* and *ViewPoint PC-60* negotiate increasingly improved data transfer formats. Newer private formats remove most of the previous limitations and provide even more, easy to use, interface capabilities.

- Public Format, *EYEDATA_29*

RemoteLink receives data from the *ViewPoint EyeTracker* in packets. The default data format is *EYEDATA_29* that is the same standard for both the Macintosh and the PC60 versions of the *ViewPoint EyeTracker*. Keeping this as the default allows this Windows version of *RemoteLink* to receive data from either platform. However this format does have some limitations.

A major limitation is that this format does not provide the DLL on the *RemoteLink* computer with all of the data that is available on the *ViewPoint EyeTracker* computer. It provides only: (x,y) position of gaze, (w,h) pupil dimensions, a region of interest (ROI) code for ten ROIs, delta time since the last data packet, and a packet index number that allow confirmation that no data packets are lost. See the *ViewPoint EyeTracker* documentation for more information about the SerialPort data packet formats, etc.

Note: Only unsmoothed position of gaze data is sent in this packet format. The data may be different on the two machines unless the *ViewPoint EyeTracker* slider control Smoothing Points is set to one (1 means no smoothing).

16.11.3 Increasing the Priority of RemoteLink

It has been determined that while running applications like *Presentation*, the priority of *RemoteLink* must be increased or *RemoteLink* will hang. The priority is now automatically increased to *Above Normal* at startup. The user may further adjust the priority in one of two ways:

1. Using the menu item: *Interface > Priority > Above Normal*
2. Using the *Windows Task Manager*:
 - (a) launch the *Windows Task Manager* by pressing: Ctrl-Alt-Delete
 - (b) select the *Process* tab,
 - (c) select *RemoteLink.exe*,
 - (d) right click for pop-up menu,
 - (e) select Set Priority > Above Normal

Chapter 17. ViewPoint Interface: GUI, SDK, CLI

17.1 General

This Chapter provides a description of each GUI controls and the equivalent CLI s 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 interface (CLI) string. The GUI control values can be saved in a Settings file that consists of the CLI strings (keyTerms and parameters), so control values can be easily loaded at any time. The CLI 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 CLI strings as well as providing access to data and other information.

17.1.1 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
```

17.1.2 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 within 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.

17.1.3 Case insensitive CLI strings

The CLI 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.

17.1.4 Boolean Toggle

All CLI s that accept BoolValue arguments (e.g.: True, False, On, Off) can also accept the argument "Toggle", e.g., "dataFile_Pause Toggle", that changes from the current state to the opposite state.

17.1.5 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.



17.2 Data Files

17.2.1 Open Data File with Randomly Generated Name

GUI:	File > Data > Unique Data File	^U
CLI :	<code>dataFile_NewUnique</code>	
SDK:	<code>VPX_DataFile_NewUnique();</code>	

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).

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.

See also:

setPath
dataFile_NewUniqueExtension

See 11.3 Data File Format for a description of the format of the data file.

```
VPX_SendCommand( "dataFile_NewUnique" );
VPX_DataFile_NewUnique();
```

17.2.2 Specify NewUnique Data File Extension

GUI:	-none
CLI :	<code>dataFile_NewUniqueExtension</code> <i>fileExtensionString</i>
SDK:	-none-
Default:	.txt

Specifies the file type extension that is appended when performing the dataFile_NewUnique operation. This specification does *not* affect the format of the data inside the file. This specification does *not* affect the dataFile_NewName command.

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 *Microsoft Excel* or *Microsoft Works* spreadsheet packages.

See section: 17.1.2 Quoting strings with white spaces

HINT: You can use the extension string to append a group name, e.g., “_drug_C_.wks” or “_MondayData.txt”,

v.2.8.1.12 CLI added.

```
dataFile_NewUniqueExtension " drug group X .xls"
VPX_SendCommand( "dataFile_NewUniqueExtension .txt" );
VPX_SendCommand( "dataFile_NewUniqueExtension \" patient_%d .txt\" ", pid );
```



17.2.3 Open a Data File and Specify a File Name		
GUI:	File > Data > New Data File ...	^N
CLI :	<code>dataFile_NewName <i>fileNameString</i></code>	
SDK:	-none-	
<p>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.</p> <p>A file type extension must be included in the string otherwise none will appear, however an extension may be added later at the operating system level.</p> <p>The <code>dataFile_NewUniqueExtension</code> specification does <i>not</i> affect this command.</p> <p>See section: 17.1.2 Quoting strings with white spaces</p> <p>The Data line in the Status window reports the current state of data recording.</p>		
<pre>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);</pre>		

17.2.4 Insert a String into the Data File		
GUI:	-none-	
CLI :	<code>dataFile_InsertString <i>string</i></code>	
SDK:	-none-	
<p>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.</p> <p>See section: 17.1.2 Quoting strings with white spaces</p> <p>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 <code>dataFile_AsynchStringData</code> The default is asynchronous.</p> <p>When called more frequently than data is saved, and the <code>dataFile_AsynchStringData</code> is set to false, then the strings are concatenated. The string separator is the tab character.</p> <p>See also:</p> <ul style="list-style-type: none"><code>dataFile_InsertMarker</code><code>dataFile_AsynchStringData</code>		
<pre>VPX_SendCommand("dataFile_InsertString showingPictureOfCat"); VPX_SendCommand("dataFile_InsertString \" Showing picture of a cat.\" "); VPX_SendCommand("dataFile_InsertString \"%s\" ", userString);</pre>		



17.2.5 Insert a Marker into the Data File

GUI:	-none-
CLI :	<code>dataFile_InsertMarker <i>DataMarker</i></code> <i>DataMarker</i> : Any single byte ascii character.
SDK:	<code>VPX_DataFile_InsertMarker (<i>DataMarker</i>);</code>
<p>Insert the specified ASCII character into the data file. This can be used for data synchronization coding, or patient responses, etc.</p> <p>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 <code>dataFile_AsynchMarkerData Yes</code> The default is synchronous.</p> <p>Where exact timing of the markers is critical, you may achieve better performance using the specific SDK function directly, rather than sending the CLI string that must first be processed by the parser.</p> <p>Try the program <code>~/ViewPoint/ExtraApps/DataMarker.exe</code> provides an easy way to manually insert markers into the data file in real-time.</p> <p><i>Note:</i></p> <p>Currently, non-printable ascii characters are not filtered out, so be careful if you specify such characters as: tab, bell, backspace, line-feed, etc.</p> <p><i>See also:</i></p> <p><code>dataFile_InsertString</code> <code>dataFile_AsynchMarkerData</code></p>	
<code>dataFile_InsertMarker K</code> <code>VPX_SendCommand ("dataFile_InsertMarker K ")</code> <code>VPX_SendCommand ("dataFile_InsertMarker %c ", theMarker)</code> <code>VPX_DataFile_InsertMarker('K')</code>	



<h3>17.2.6 Insert a User Defined Data Tag into the Data File</h3>	
GUI:	-none-
CLI :	<code>dataFile_InsertUserTag <i>UserTag string</i></code>
SDK:	-none-
<p>This allows users to insert data from their own sources into a data file with their own specified tag. The tag appears in column 1 of the data file. The insertion is done asynchronously with respect to the eye movement data records and the insertions are uniquely time stamped. Tag identifiers must be in the range 800-899.</p> <p>See: Section 11.3.2 on page 54.</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>	

<h3>17.2.7 Specifies asynchronous or synchronous string data</h3>	
GUI:	File > Data > Asynchronous String Data
CLI :	<code>dataFile_AsynchStringData <i>BoolValue</i></code> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-none-
Default:	Yes
<p>Specifies whether to insert string 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 11.3.2 on page 54.</p> <p>See also:</p> <p>dataFile_InsertString</p>	
<pre>VPX_SendCommand("datafile_ AsynchStringData No");</pre>	



17.2.8 Specify asynchronous or synchronous marker data	
GUI:	File > Data > Asynchronous Marker Data
CLI :	<code>dataFile_AsynchMarkerData</code> <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-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 11.3.2 on page 54.</p> <p>See also: dataFile_InsertMarker</p>	
<code>VPX_SendCommand("datafile_AsynchMarkerData Yes");</code>	

17.2.9 Specify asynchronous or synchronous head tracker data	
GUI:	File > Data > Asynchronous Head Tracker Data
CLI :	<code>dataFile_AsynchHeadData</code> <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-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 11.3.2 on page 54.</p> <p>See also: headTrackerConnect</p>	
<code>VPX_SendCommand("dataFile_AsynchHeadData yes");</code>	



17.2.10	Specify data file start time
GUI:	File > Data > Start Data File Time at Zero
CLI :	<code>dataFile_startFileTimeAtZero</code> <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-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:</p> <p>VPX_GetPrecisionDeltaTime</p>	
<code>VPX_SendCommand("datafile_StartFileTimeAtzero No");</code>	



17.2.11 Store smoothed or unsmoothed data	
GUI:	File > Data > Store Smoothed Data
CLI :	dataFile_StoreSmoothedData <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-none-
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>See also: smoothingPoints, smoothingMethod</p>	
VPX_SendCommand("datafile_StoreSmoothedData Yes");	

17.2.12 Include Raw Eye Data	
GUI:	File > Data > Include Raw (unmapped EyeSpace data)
CLI :	dataFile_includeRawData <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-none-
Default:	No
<p>Specifies whether to store raw eye data.</p>	
VPX_SendCommand("datafile_includeRawData Yes");	



17.2.13 Specify whether to use buffering (DEPRECATED)	
GUI:	-removed-
CLI :	dataFile_UseBuffering <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	VPX_DataFile_Buffering(<i>bool tf</i>);
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>	

17.2.14 Pause writing of data to file	
GUI:	File > Data > Pause Data Capture (toggle) ^P
CLI :	dataFile_Pause dataFile_Resume <i>No arguments.</i>
SDK:	VPX_DataFile_Pause(<i>bool tf</i>);
<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>	



17.2.15 Close Data File		
GUI:	File > Data > Close Data File	^W
CLI :	<code>dataFile_Close</code>	
SDK:	<code>VPX_DataFile_Close();</code>	
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		
<code>VPX_SendCommand("dataFile_Close");</code> <code>VPX_DataFile_Close();</code>		

17.2.16 Close Data File and Open in Post-Hoc Analysis tool		
GUI:	File > Data > Close & Analyze Data File	Alt-Shift-W
CLI :	<code>dataFile_CloseAndAnalyze</code>	
SDK:	<code>-none-</code>	
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. The Data line in the Status window reports the current state of data recording.		
<code>VPX_SendCommand("dataFile_CloseAndAnalyze");</code> <code>VPX_closeAndAnalyzeDataFile();</code>		

17.3 Stimulus Images

17.3.1 Load Stimulus Image into the Stimulus window		
GUI:	File > Images > Load Image	^I
CLI :	<code>stimulus_LoadImageFile <i>filename</i></code>	
SDK:	<code>-none-</code>	
Loads the selected image into the Stimulus window See section: 17.1.2 Quoting strings with white spaces See also: stimulusGraphicsOptions +Image		
<code>VPX_SendCommand("stimulus_LoadImageFile catPicture.bmp");</code> <code>VPX_SendCommand("stimulus_LoadImageFile \"second cat picture .bmp\" ");</code> <code>VPX_SendCommand("stimulus_LoadImageFile %s", bitmapFileName);</code>		



17.3.2 Specifies how to display the currently loaded stimulus image

GUI	Stimuli > Image Shape > shape type
CLI :	<code>stimulus_ImageShape</code> <i>ShapeType</i> <i>ShapeType</i> : Actual, Centered, Fit, Isotropic
SDK:	-none-
Default:	Fit

Specifies how the bitmap image is to be displayed in the Stimulus windows.

A = Actual : Displays the image in the two windows at actual size.

C = Centered : Displays the image actual size and centered in the windows.

F = Fit : Displays the image stretched un-equally to fit the window.

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.

See also:

`stimulus_BackgroundColor`

`VPX_STATUS_StimulusImageShape`

`VPX_SendCommand("stimulus_ImageShape Isotropic");`

17.3.3 Specify a background "matting" color for the stimulus window

GUI:	Stimuli > Background Color
CLI :	<code>stimulus_BackgroundColor</code> <i>ColorValue</i> <i>ColorValue</i> : Red 0-255, Green 0-255, Blue 0-255
SDK:	-none-

Sets 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.

Workarounds: The window may need to be minimized and reopened to show the color changes.

See also:

`stimulus_ImageShape`

Eg.set to BRIGHT RED use: `stimulus_BackgroundColor 255 0 0`

`VPX_sendCommand ("stimulus_BackgroundColor 255 135 075");`



<h3>17.3.4 Stereoscopic Display (Side-by-side)</h3>	
GUI:	-none-
CLI :	stereoDisplay <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-none-
Specifies that the stimulus image should be displayed as two identical side-by-side images in the Stimulus window and in the GazeSpace window; and that the calibration stimulus points should be displayed in stereo pairs in each side-by-side frame. New: 2.8.3.40	
VPX_sendCommand ("stereoDisplay YES");	

<h3>17.3.5 Play specified Sound file</h3>	
GUI:	-none-
CLI :	stimulus_PlaySoundFile <i>soundFileName</i>
SDK:	-none-
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". See section: 17.1.2 Quoting strings with white spaces This feature may cause a media/audio player to open, if this happens, check your computer settings. If this problem persists, try using midi notes.	
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);	

17.4 PictureList

<h3>17.4.1 Initialize Picture List</h3>	
GUI:	-none-
CLI :	pictureList_Init
SDK:	-none-
Initializes the list for stimulus images, making it ready for new names to be entered.	
pictureList_Init VPX_SendCommand("pictureList_Init");	



<h3>17.4.2 Add List of Image Names to PictureList</h3>		
GUI:	-none-	
CLI :	<code>pictureList_AddName <i>imageFileName</i></code>	
SDK:	-none-	
Adds an image file name to the picture list.		
<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>		
<h3>17.4.3 Randomize List of Images in the PictureList</h3>		
GUI:	File > Images > Picture List > Randomize PictureList	
CLI :	<code>pictureList_Randomize</code>	
SDK:	-none-	
Randomizes the pointers in the picture list. Repeat this to re-randomize.		
<pre>VPX_SendCommand("pictureList_Randomize");</pre>		
<h3>17.4.4 Move to Next Image in the PictureList</h3>		
GUI:	File > Images >PictureList>Next PictureList Image	^F12 (default)
CLI :	<code>pictureList_ShowNext</code>	
SDK:	-none-	
Moves to the next file pointer in the picture list.		
<pre>VPX_SendCommand ("pictureList_ShowNext");</pre>		
<h3>17.4.5 Move to Start of Images in Picture List</h3>		
GUI:	File > Images > PictureList > Restart PictureList	
CLI :	<code>pictureList_Restart</code>	
SDK:	-none-	
Re-sets the pointer to the first image in the list. Does not un-randomize if the list has been randomized.		
<pre>VPX_SendCommand ("pictureList_Restart");</pre>		



17.5 Controls window: EyeImage

<h3>17.5.1 Specify Mapping Feature</h3>	
GUI:	Controls Window, Video Image tab, Feature Method pull down menu
CLI :	mappingFeature <i>Method</i> //sets Eye A EyeA:mappingFeature <i>Method</i> EyeB:mappingFeature <i>Method</i> <i>Methods</i> : Pupil, Glint, Vector, Manual, SlipComp, Pattern
SDK:	int VPX_SetFeatureMethod (int <i>featureMethod</i>) int VPX_SetFeatureMethod2(VPX_EyeType <i>eyn</i> , int <i>featureMethod</i>) <i>featureMethod</i> : PUPIL_ONLY_Method, GLINT_ONLY_Method, VECTOR_DIF_Method
Default:	Pupil
Controls what type of mapping will be done from EyeSpace to GazeSpace for the specified eye. Return value: 0=false, 1=true, otherwise -1 if invalid value for eye	
VPX_SendCommand ("mappingFeature method"); // Eye A VPX_SendCommand ("EyeA:mappingFeature method"); VPX_SendCommand ("EyeB:mappingFeature method");	

<h3>17.5.2 AutoThreshold</h3>	
GUI:	Controls window, VideoImage tab, Threshold: Autothreshold button
CLI :	autoThreshold // Eye A EyeA:autoThreshold EyeB:autoThreshold autoThreshold_Start
SDK:	int VPX_AutoThreshold(); int VPX_AutoThreshold2(VPX_EyeType <i>eye</i>); <i>VPX_EyeType</i> : Eye_A, Eye_B
Automatically sets desirable glint and pupil threshold levels for the specified eye.	
VPX_SendCommand ("autoThreshold"); // Eye A VPX_SendCommand ("EyeA:autoThreshold"); VPX_SendCommand ("EyeB:autoThreshold"); VPX_AutoThreshold2(EYE_A);	



<h3>17.5.3 Positive Lock Tracking</h3>	
GUI:	Controls Window, Video Image Tab, Threshold Group, Positive-Lock Threshold-Tracking
CLI :	<code>positiveLock BoolValue //sets Eye A</code> <code>EyeA:positiveLock BoolValue</code> <code>EyeB:positiveLock BoolValue</code> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK:	<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:	On
Continuous automatic feature threshold adjustment for the specified eye.	
<code>VPX_SendCommand ("positiveLock on"); // Eye A</code> <code>VPX_SendCommand ("EyeA:positiveLock on");</code> <code>VPX_SendCommand ("EyeBpositiveLock on");</code>	

<h3>17.5.4 Adjust Pupil Threshold Slider</h3>	
GUI:	Controls window, EyeImage tab, Pupil Threshold slider
CLI :	<code>pupilThreshold NormalizedValue // Eye A</code> <code>EyeA:pupilThreshold NormalizedValue</code> <code>EyeB:pupilThreshold NormalizedValue</code> <i>NormalizedValue: a floating point number in range 0.0 to 1.0</i> <code>pupilThreshold EyeType NormalizedValue</code> <i>EyeType: Eye_A, Eye_B</i>
SDK:	<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:	0.25 , but autothreshold at startup may reset this
Sets the image intensity threshold for the specified eye 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. See also: autoThreshold	
<code>VPX_SendCommand ("pupilThreshold 0.7"); // eye A</code> <code>VPX_SendCommand ("EyeA:pupilThreshold 0.7");</code> <code>VPX_SendCommand ("EyeB:pupilThreshold 0.7");</code>	



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

17.5.6 Adjust Video Image Brightness	
GUI:	Controls window, EyeImage tab, Brightness slider
CLI :	videoImageBrightness <i>NormalizedValue</i> // sets EyeA EyeA:videoImageBrightness <i>NormalizedValue</i> EyeB:videoImageBrightness <i>NormalizedValue</i> <i>NormalizedValue</i> : a floating point number in range 0.0 to 1.0
SDK:	-none- The SDK function has been deprecated. Use VPX_SendCommand int VPX_SetImageBrightness (float <i>normalValue</i>) int VPX_SetImageBrightness2(VPX_EyeType <i>eyn</i> , float <i>normalVal</i>) <i>normalValue</i> : a floating point number in range 0.0 to 1.0
Default:	- varies -
Sets the video image brightness levels of the specified eye, normalized from 0.0 to 1.0.	
VPX_SendCommand("videoImageBrightness 0.5"); //sets EyeA VPX_SendCommand("EyeA:videoImageBrightness 0.5"); //sets EyeA brightness VPX_SendCommand("EyeB:videoImageBrightness 0.5"); //sets EyeB brightness	



17.5.7 Adjust Video Image Contrast	
GUI:	Controls window, EyeImage tab, Contrast slider
CLI :	<code>videoImageContrast <i>normalValue</i> // sets EyeA</code> <code>EyeA:videoImageContrast <i>normalValue</i></code> <code>EyeB:videoImageContrast <i>normalValue</i></code> <i>normalValue</i> : a floating point number in range 0.0 to 1.0
SDK:	-none- The SDK function has been deprecated. Use VPX_SendCommand <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 of the specified eye, normalized from 0.0 to 1.0.	
<code>VPX_SendCommand("videoImageContrast 0.7"); //sets Eye A</code> <code>VPX_SendCommand("EyeA:videoImageContrast 0.7");</code> <code>VPX_SendCommand("EyeB:videoImageContrast 0.7");</code>	

17.5.8 Dynamically Optimize Brightness and Contrast Settings	
GUI:	Controls window, EyeImage tab, AutoImage checkbox
CLI :	<code>videoAutoImage <i>BoolValue</i></code> <code>EyeA:videoAutoImage <i>BoolValue</i></code> <code>EyeB:videoAutoImage <i>BoolValue</i></code>
SDK:	-none-
Default:	On
Continuously attempts to set the video image brightness and contrast levels to optimal values for the specified eye. <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"); // sets Eye A</code> <code>VPX_SendCommand("EyeA:videoAutoImage On");</code> <code>VPX_SendCommand("EyeA:videoAutoImage On");</code>	

17.5.9 Adjust Pupil Scan Density	
GUI:	Controls window, EyeImage tab, Pupil Scan Density slider
CLI :	<p>pupilScanDensity <i>FloatValue</i> // sets Eye A</p> <p>EyeA:pupilScanDensity <i>FloatValue</i></p> <p>EyeB:pupilScanDensity <i>FloatValue</i></p> <p><i>FloatValue</i> : integer in range 1 to 20</p>
SDK:	<p>VPX_SetPupilResolution (float <i>resolution</i>);</p> <p>VPX_SetPupilResolution2 (VPX_EyeType eye, float <i>resolution</i>);</p> <p>VPX_EyeType : Eye_A, Eye_B</p> <p><i>resolution</i> : use only whole value in range 1 to 20</p>
Default:	7
<p>The value specifies the pixel sampling interval for the threshold segmentation operation for the specified eye. 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 CLI : minimumPupilScanDensity.</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:</p> <p>minimumPupilScanDensity</p>	
<pre>VPX_SendCommand ("pupilScanDensity 5"); // seys Eye A VPX_SendCommand ("EyeA:pupilScanDensity 5"); VPX_SendCommand ("EyeB:pupilScanDensity 5");</pre>	



17.5.10	Override Pupil Scan Density Minimum
GUI:	-none-
CLI :	minimumPupilScanDensity <i>DensityIndex</i> // sets Eye A EyeA:minimumPupilScanDensity <i>DensityIndex</i> EyeB:minimumPupilScanDensity <i>DensityIndex</i> <i>DensityIndex: Integer in range 3 to 20, depending upon max density chosen.</i>
SDK:	-none-
Default:	7
Overrides the minimum pupil scan density of the specified eye 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.	
VPX_SendCommand ("minimumPupilScanDensity 12"); // sets Eye A VPX_SendCommand ("EyeA:minimumPupilScanDensity 12"); VPX_SendCommand ("EyeB:minimumPupilScanDensity 12");	



17.5.11	Adjust Glint Scan Density
GUI:	Controls window, EyeImage tab, Glint / Scan Density slider
CLI :	<code>glintScanDensity <i>IntValue</i> // sets Eye A</code> <code>EyeA:glintScanDensity <i>IntValue</i></code> <code>EyeB:glintScanDensity <i>IntValue</i></code> <i>IntValue</i> : integer in range 1 to 20
SDK:	<code>VPX_SetGlintResolution (<i>float resolution</i>);</code> <code>VPX_SetGlintResolution2 (<i>VPX_EyeType eye</i>, <i>float resolution</i>);</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 for the specified eye. The value 1 indicates to sample 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.</p> <p>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 CLI : 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:</p> <ul style="list-style-type: none">minimumGlintScanDensityglintSegmentationMethod	
<code>VPX_SendCommand ("glintScanDensity 5"); // sets Eye A</code> <code>VPX_SendCommand ("EyeA:glintScanDensity 5");</code> <code>VPX_SendCommand ("EyeB:glintScanDensity 5");</code>	



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

17.6 EyeCamera Window

17.6.1 Adjust Pupil Scan Area	
GUI:	EyeCamera window , EyeCamera toolbar , Top Button
CLI :	pupilScanArea <i>L T R B</i> // sets eye A EyeA:pupilScanArea <i>L T R B</i> EyeB:pupilScanArea <i>L T R B</i> <i>L T R B: Normalized floating point values for the corners.</i>
SDK:	<code>VPX_SetPupilScanArea (VPX_RealRect rr);</code>
Default:	0.200 0.200 0.800 0.800
<p>Defines scan area for the pupil of the specified eye. 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"); // sets eye A VPX_SendCommand ("EyeA:pupilScanArea 0.3 0.2 1.0 0.4"); VPX_SendCommand ("EyeB:pupilScanArea 0.3 0.2 1.0 0.4");</pre>	



<h3>17.6.2 Specify Pupil Scan Area Shape</h3>	
GUI:	-none-
CLI :	pupilScanShape <i>ScanShapeType</i> // sets Eye A EyeA:pupilScanShape <i>ScanShapeType</i> EyeB:pupilScanShape <i>ScanShapeType</i> <i>ScanShapeType</i> : Rectangle, Ellipse.
SDK:	-none-
Default:	Elliptical
Specifies whether to change the scan area for the pupil to either rectangular or elliptical for the specified eye. Elliptical scan area is effective at eliminating dark spots that the software interprets as a pupil. Elliptical Scan. The over lay graphic	
VPX_SendCommand("pupilScanShape Ellipse"); // sets Eye A VPX_SendCommand("EyeA:pupilScanShape Ellipse"); VPX_SendCommand("EyeB:pupilScanShape Ellipse");	

<h3>17.6.3 Pupil and Glint oval fit constraints</h3>	
GUI:	-none-
CLI :	pupilConstrained <i>BoolValue</i> // Sets Eye A EyeA:glintConstrained <i>BoolValue</i> EyeB:glintConstrained <i>BoolValue</i>
SDK:	-none-
Default:	Yes (in most versions)
Controls whether or not the pupil (glint) oval fit is allowed to extend beyond the scan area rectangle in the EyeSpace window.	
VPX_SendCommand("pupilConstrained True");	



17.6.4 Define Glint Scan Area	
GUI:	EyeCamera window , EyeCamera toolbar , Third Button to enable Drag mouse in window to define the glint scan area rectangle.
CLI :	glintScanSize X Y // Eye A EyeA:glintScanSize X Y EyeB:glintScanSize X Y
SDK:	VPX_SetGlintScanSize (VPX_RealPoint rp);
Default:	0.400 0.200
Defines scan area for the glint of the specified eye. The two values are the normalized floating point X,Y size values of the scan rectangle.	
VPX_SendCommand ("glintScanSize -0.4 1.3"); // sets Eye A VPX_SendCommand ("EyeA:glintScanSize -0.4 1.3"); VPX_SendCommand ("EyeB:glintScanSize -0.4 1.3");	

17.6.5 Define Offset of Glint Scan Area Relative to the Pupil	
GUI:	-none-
CLI :	glintScanOffset X Y // sets eye A EyeA:glintScanOffset X Y EyeB:glintScanOffset X Y <i>X Y : Normalized floating point coordinates.</i>
SDK:	VPX_SetGlintScanOffset (VPX_RealPoint rp);
Default:	0.010 0.080
Defines offset of the glint scan area relative to the center of the pupil for the specified eye. The two values are the normalized coordinates of the offset vector from the center of the pupil. See also: video_yokedGlint NO glintScanUnYokedOffset	
VPX_SendCommand ("glintScanOffset 0.4 0.3"); // sets eye A VPX_SendCommand ("EyeA:glintScanOffset 0.4 0.3"); VPX_SendCommand ("EyeB:glintScanOffset 0.4 0.3");	



17.6.6 Unyoke Glint Scan Area from the Pupil	
GUI:	-none-
CLI :	video_yokedGlint <i>BoolValue</i> //sets EyeA EyeA:video_yokedGlint <i>BoolValue</i> EyeB:video_yokedGlint <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-none-
Default:	Yes
Allows the glint scan area to be unyoked from the pupil of the specified eye. i.e. the glint scan area does not move with the pupil. Special applications only. See also: glintScanUnYokedOffset glintScanOffset	
VPX_SendCommand("video_yokedGlint On"); // Eye A VPX_SendCommand("EyeA:video_yokedGlint On"); VPX_SendCommand("EyeB:video_yokedGlint On");	

17.6.7 Define offset of Unyoked Glint Scan Area	
GUI:	-none-
CLI:	glintScanUnYokedOffset <i>X Y</i> // Sets Eye A EyeA:glintScanUnYokedOffset <i>X Y</i> EyeA:glintScanUnYokedOffset <i>X Y</i> <i>X Y</i> : <i>Normalized floating point coordinates.</i>
SDK:	VPX_SetGlintScanUnyokedOffset(<i>VPX_RealPoint rp</i>);
Defines offset of the glint scan area relative to the upper left hand corner of the EyeCamera window for the specified eye. The two values are the normalized coordinates of the offset vector. See also: video_yokedGlint	
VPX_SendCommand ("glintScanUnYokedOffset 0.1 -3.0"); VPX_SendCommand ("EyeA:glintScanUnYokedOffset 0.1 -3.0"); VPX_SendCommand ("EyeB:glintScanUnYokedOffset 0.1 -3.0");	



17.6.8 Toggle Show Treshold Dots On / Off	
GUI:	Button on EyeCamera window
CLI :	showThresholdDots <i>BoolValue</i> EyeA:showThresholdDots <i>BoolValue</i> EyeB:showThresholdDots <i>BoolValue</i> // sets Eye A <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	VPX_Video_ShowThresholdDots (<i>bool tf</i>);
Default:	Yes
Specifies whether the image segmentation dots are displayed in the EyeCamera window for the specified eye.	
VPX_SendCommand ("showThresholdDots No"); // sets eye A VPX_SendCommand ("EyeA:showThresholdDots No"); VPX_SendCommand ("EyeB:showThresholdDots No");	

17.6.9 Specify EyeImage Overlay Graphics sent to layered application (EXPERIMENTAL)	
Control or Menu:	-none-
CLI :	vpx_EyeCameraImageOverlays <i>StringArg</i> <i>StringArg</i> : +Eye_A -Eye_A +Eye_B -Eye_B
SDK:	VPX_SetEyeImageOverlays(VPX_EyeType eye, bool tf); <i>VPX_EyeType</i> : Eye_A, Eye_B
EXPERIMENTAL : This functionality is under development and may not be stable over future versions.	
This only affects the overlay graphics in the remote eye image that is sent to a layered SDK application with the SDK function VPX_SetEyeImageWindow. 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 showThresholdDots	
Note carefully: the CLI and the SDK names are spelled differently.	
See also:	
VPX_SetEyeImageWindow showThresholdDots	
VPX_SendCommand ("vpx_EyeCameraImageOverlays off"); VPX_SetEyeImageOverlays(Eye_A, 1);	



17.6.10	EyeCamera Tool Bar Display
GUI:	-none-
CLI :	videoToolBar <i>BoolValue</i> // sets Eye A EyeA:videoToolBar <i>BoolValue</i> EyeB:videoToolBar <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	VPX_Video_ShowEyeCameraToolBar (<i>bool tf</i>);
Default:	Yes
Specifies whether to display eye camera tool bar for the specified eye.	
VPX_SendCommand("videoToolBar off"); // sets eye A VPX_SendCommand("EyeA:videoToolBar off"); VPX_SendCommand("EyeB:videoToolBar off");	

17.7 Slip Compensation

17.7.1 Slip Compensation Mapping Mode	
GUI:	Controls window > EyeImage tab > Feature Method Pull Down menu > SlipCompensation
CLI :	mappingFeature <i>Method</i> //sets Eye A EyeA:mappingFeature <i>Method</i> EyeB:mappingFeature <i>Method</i> <i>Methods</i> : Pupil, Glint, Vector, Manual, SlipComp, Pattern
SDK:	int VPX_SetFeatureMethod (int <i>featureMethod</i>) int VPX_SetFeatureMethod2(VPX_EyeType <i>eyn</i> , int <i>featureMethod</i>) <i>featureMethod</i> : PUPIL_ONLY_Method, GLINT_ONLY_Method, VECTOR_DIF_Method
Default:	Pupil
Controls what type of mapping will be done from EyeSpace to GazeSpace for the specified eye. Slip Compensation provides a means to compensate for HMD slippage. Return value: 0=false, 1=true, otherwise -1 if invalid value for eye	
VPX_SendCommand("mappingfeature SlipComp"); // sets eye A VPX_SendCommand("EyeA: mappingfeature SlipComp"); VPX_SendCommand("EyeB: mappingfeature SlipComp");	



17.7.2 Slip Compensation Speed

GUI: **None**

CLI : SlipComp_speed *floatValue* // sets Eye A
EyeA: SlipComp_speed *floatValue*
EyeB: SlipComp_speed *floatValue*
floatValue: in range [0.005 1.0]

SDK: Use VPX_SendCommand

Default: **0.03**

This is the speed at which the calculated error value due to slip is modified. A value of 1.0 causes immediate response to any error -- it will not filter vibrations and it will not filter transient glint loss, which are among the primary advantages of the SlipCompensation method. A value of 0.5 would use half of the current value and half of the previous average error based on the exponential moving average (EMA) algorithm. A small parameter value is suitable for very slow slip. Because the EMA is recalculated with every new eye image, even a small value can produce an effect reasonably quickly. To get a feel for the amount of correction, try moving (slipping) the EyeCamera deliberately while fixating on the position of gaze overlay spot and see how quickly it recovers. Also the SlipComp PenPlot displays show how the value varies over time.

See also: slipComp_yGain , slipComp_yGain

```
VPX_SendCommand( "videoToolBar off"); // sets eye A  
VPX_SendCommand( "EyeA:videoToolBar off");  
VPX_SendCommand( "EyeB:videoToolBar off");
```

17.7.3 Slip Compensation X-Gain

GUI: **None**

CLI : SlipComp_xGain *floatValue* // sets Eye A
EyeA: SlipComp_xGain *floatValue*
EyeB: SlipComp_xGain *floatValue*
floatValue: in range: Real

SDK: Use VPX_SendCommand

Default: **1, i.e. no gain**

Given a certain (slowly varying) calculated slip error, this controls how much that error value affects the gaze point in the horizontal. Default is 1.0, i.e. no gain as HMD slippage is typical in the vertical direction.

See also: slipComp_speed, slipComp_yGain



17.7.4 Slip Compensation Y-Gain	
GUI:	None
CLI :	SlipComp_yGain <i>floatValue</i> // sets Eye A EyeA: SlipComp_yGain <i>floatValue</i> EyeB: SlipComp_yGain <i>floatValue</i> <i>floatValue</i> : in range: Real
SDK:	Use VPX_SendCommand
Default:	1.1, i.e. slight gain
Given a certain (slowly varying) calculated slip error, this controls how much that error value affects the gaze point in the vertical. See also: slipComp_speed, slipComp_xGain	

17.8 Video related controls

17.8.1 Specify EyeCamera Video Input Standard

GUI: **EyeCamera window > monitor icon > Video Standard > NTSC, PAL, SECAM**

CLI :
 videoStandard *Option* // Eye A
 EyeA:videoStandard *Option*
 EyeB:videoStandard *Option*
Option: any one of: NTSC, PAL, SECAM

SDK: -none-

Default: **NTSC upon first run, but saved in preferences file thereafter.**

Specifies which video mode to use for the specified eye.

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 ViewPoint is launched.

```
VPX_SendCommand( "videoStandard option"); // Eye A
VPX_SendCommand( "videoStandard option");
VPX_SendCommand( "videoStandard option");
```

17.8.2 Specify SceneCamera Video Input Standard

GUI: -none-

CLI :
 scene_videoStandard *Option*
Option: any one of: NTSC, PAL, SECAM

SDK: -none-

Default: **NTSC**

The SceneCamera video standard (signal standard) can now be changed using the following command:

```
scene_videoStandard <format>
  where <format> is one of: NTSC, PAL, SECAM
```

Implemented 2.8.5.017

```
VPX_SendCommand( "scene_VideoStandard option");
```




17.8.3 Specify Tracking Operation Mode	
GUI:	EyeCamera window > Monitor Icon > Mode > Setup, Precision, Speed
CLI :	videoMode <i>ProcessingMode</i> // sets Eye A EyeA:videoMode <i>ProcessingMode</i> EyeB:videoMode <i>ProcessingMode</i> <i>ProcessingMode</i> : Setup, Precision, Speed, Speed2
SDK:	VPX_Video_Mode (VideoProcessingMode mode); Setup_Mode, HighPrecision_Mode, HighSpeed_Mode
Default:	Setup
Specifies which operation mode to use for the specified eye.	
VPX_SendCommand("videoMode Precision"); // sets Eye A VPX_SendCommand("EyeA:videoMode Precision"); VPX_SendCommand("EyeB:videoMode Precision");	

17.8.4 Specify Dark or Bright Pupil Tracking	
GUI:	EyeCamera window > Monitor Icon > Pupil Type > Dark Pupil, Bright Pupil
CLI :	pupilType <i>PupilType</i> // Eye A EyeA:pupilType <i>PupilType</i> EyeB:pupilType <i>PupilType</i> <i>PupilType</i> : Dark, Bright
SDK:	int VPX_SetPupilType (<i>PupilMethod</i>); int VPX_SetPupilType2(VPX_EyeType eyn, int method); DARK_PUPIL_Method, BRIGHT_PUPIL_Method
Specifies dark or bright pupil tracking for the specified eye.	
VPX_SendCommand("pupilType dark"); // Eye A VPX_SendCommand("EyeA:pupilType dark"); VPX_SendCommand("EyeB:pupilType dark");	

17.8.5 Specify Pupil Segmentation Method	
GUI:	EyeCamera window > Monitor Icon > Pupil Segmentation Method > Centroid, Oval Fit, Ellipse
CLI :	pupilSegmentationMethod <i>Method</i> // Eye A EyeA:pupilSegmentationMethod <i>Method</i> EyeB:pupilSegmentationMethod <i>Method</i> <i>Method</i> : Centroid, OvalFit, Ellipse
SDK:	send command line instruction via VPX_SendCommand routine.
Default:	OvalFit (for older versions), FitEllipse (for newer versions)
Specifies which pupil segmentation method to use for the specified eye. The centroid	



is obtained first in all cases, but OvalFit or Ellipse perform additional image processing and fitting.
Note1: the pupil width and pupil aspect ratio will not be available unless either OvalFit or Ellipse is selected.

Note2: the calculation for pupil width and pupil aspect ratio are different for these two methods.

OvalFit looks only at the horizontal width.

Ellipse uses the length of the major axis of the rotated ellipse, which is far more stable for tertiary positions of gaze.

Note3: blinks are detected by examining when the pupil aspect ratio is below criterion; this calculation usually performs better with the older OvalFit than the newer Ellipse method.

Example: `VPX_SendCommand("pupilsegmentationMethod OvalFit");`

```
VPX_SendCommand( "pupilsegmentationMethod Centroid" ); // Eye A
```

```
VPX_SendCommand( "EyeA:pupilsegmentationMethod Centroid" );
```

```
VPX_SendCommand( "EyeB:pupilsegmentationMethod Centroid" );
```

17.8.6 Specify Glint Segmentation Method

GUI: **EyeCamera window > Monitor Icon > Glint Segmentation Method > Centroid, Oval Fit**

CLI : `glintSegmentationMethod Method // Eye A`
`EyeA:glintSegmentationMethod Method`
`EyeB:glintSegmentationMethod Method`
Method: Centroid, OvalFit

SDK: -none-

Default: **OvalFit**

Specifies which glint segmentation method to use for the specified eye, either ovalFit or centroid. Actually the centroid is obtained in either case, but OvalFit performs additional image processing and fitting.

Note: changing this may also change the values of `minimumGlintScanDensity` and `glintScanDensity`.

```
VPX_SendCommand( "glintSegmentationMethod OvalFit" ); // Eye A
```

```
VPX_SendCommand( "EyeA:glintSegmentationMethod OvalFit" );
```

```
VPX_SendCommand( "EyeB:glintSegmentationMethod OvalFit" );
```



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

17.8.8 Toggle Freeze Video Image Preview On / Off	
GUI:	EyeCamera window, SnowFlake button ^F
CLI :	<code>videoFreeze</code> <i>BoolValue</i> // EyeA <code>EyeA:videoFreeze</code> <i>BoolValue</i> <code>EyeB:videoFreeze</code> <i>BoolValue</i> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK:	<code>VPX_Video_Freeze (bool tf);</code>
Default:	No
Allows freezing the video image preview, Status window display and data collection. (c.f. <code>freezeStops</code>) for the specified eye.	
VPX_SendCommand("videofreeze Yes"); // Eye A VPX_SendCommand("EyeA:videofreeze Yes"); VPX_SendCommand("EyeB:videofreeze Yes");	



17.8.9 Reset Video Capture Device	
GUI:	EyeCamera window > monitor icon> Reset EyeCamera Video Stimuli > ViewSource > Reset SceneCamera
CLI :	videoReset // Eye A EyeA:videoReset EyeB:videoReset videoReset <i>arg</i> <i>arg</i> : EyeA EyeB Scene
SDK:	-none- <code>VPX_Video_Reset (void); // deprecated</code>
Resets the specified video capture device for the specified eye or scene video.	
VPX_SendCommand("videoReset"); Eye A VPX_SendCommand("EyeA:videoReset"); VPX_SendCommand("EyeB:videoReset");	

17.9 Calibration controls

17.9.1 Start Auto-Calibration

GUI:	EyeSpace window, Auto-Calibrate / Auto-Calibrate button (toggle) ^A
CLI :	calibrationStart
SDK:	VPX_AutoCalibrate();

Starts the calibration process.

VPX_SendCommand ("calibrationStart");

17.9.2 Stop Auto-Calibration

GUI:	EyeSpace window, Auto-Calibrate / STOP Calibration button (toggle) ^K
CLI :	calibrationStop
SDK:	VPX_StopCalibration ();

Stops the calibration process before its normal completion.

VPX_SendCommand ("calibrationStop");

17.9.3 Specify Calibration Stimulus Presentation Speed

GUI:	EyeSpace window, Advanced button, Duration slider
CLI :	calibration_StimulusDuration <i>milliseconds</i> <i>milliseconds</i> : integer between 1 and 400
SDK:	VPX_SetCalibrationSpeed (int <i>milliseconds</i>); <i>milliseconds</i> : integer between 1 and 400
Default:	80 for Windows XP (depends upon the OS)

Specifies the delay in milliseconds between calibration stimulus changes (zoom rectangle decrements). *ViewPoint* 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.

The maximum duration is 200 ms, and the minimum is 1 ms.

VPX_SendCommand ("Calibration_StimulusDuration 145");



17.9.4 Specify the duration of presentation of calibration warning notice	
GUI:	EyeSpace window, Advanced button, Warning slider
CLI :	calibration_WarningTime <i>durationUnits</i> <i>durationUnits</i> : integer between 0 and 100
SDK:	-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: calibration_StimulusDuration</p> <p>The value 0 specifies no warning is to be given.</p> <p>See also: Calibration_StimulusDuration</p>	
VPX_SendCommand ("Calibration_WarningTime 15");	

17.9.5 Specifies Interval Between Presentation of Calibration Stimulus Points	
GUI:	-none-
CLI :	calibration_ISI <i>intervalUnits</i> <i>intervalUnits</i> : integer between 1 and 9
SDK:	-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: calibration_StimulusDuration.</p> <p>The value 0 specifies no ISI time.</p> <p>See also: calibration_StimulusDuration</p>	
VPX_SendCommand ("Calibration_ISI 2");	



17.9.6 Calibration Snap Mode	
GUI:	EyeSpace window, Advanced button, Snap Mode checkbox
CLI :	calibration_SnapMode BoolValue <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	VPX_set_snapCalibrationMode(bool tf);
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: calibration_Snap.</p> <p>See also:</p> <ul style="list-style-type: none">calibration_Snapcalibration_AutoIncrementcalibrationRedoPoint	
VPX_SendCommand ("calibration_SnapMode ON");	

17.9.7 RePresent in Snap Calibration Mode	
GUI:	EyeSpace window, Re-Present * (when asterisk is showing)
CLI :	calibration_Snap
SDK:	-none-
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 calibration_SnapMode.</p>	
VPX_SendCommand ("calibration_Snap");	



17.9.8 AutoIncrement Calibration Mode	
GUI:	EyeSpace window, Advanced button, Auto-Increment checkbox
CLI :	Calibration_AutoIncrement <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-none-
Default:	On
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 ");	

17.9.9 Calibration Stimulus Point Presentation Order	
GUI:	EyeSpace window, Advanced button, Presentation Order pulldown menu
CLI :	calibration_PresentationOrder <i>orderChoice</i> <i>orderChoice</i> : Sequential, Random, Custom
SDK:	-none-
This affects both auto-calibration 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");	

17.9.10 Specify Number of Calibration Stimulus Points	
GUI:	EyeSpace window, drop-down menu: 6 ... 72
CLI :	calibration_Points <i>numberOfPoints</i> <i>numberOfPoints</i> : integer from the following set: { 6, 9, 12, 16, 20, 25, 30, 36, 42, 49, 56, 64, 72 }
SDK:	VPX_SetCalibrationPoints (int <i>numberOfPoints</i>); <i>numberOfPoints</i> : integer from the following set: { 6, 9, 12, 16, 20, 25, 30, 36, 42, 49, 56, 64, 72 }
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");	



17.9.11	Specify Calibration Stimulus Point Color
GUI:	EyeSpace window, Advanced button, Set Stimulus Color button
CLI :	calibration_StimulusColor <i>ColorValue</i> <i>ColorValue: Red 0-255, Green 0-255, Blue 0-255</i>
SDK:	-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");	

17.9.12	Specify Calibration Stimulus Window Background Color
GUI:	EyeSpace window, Advanced button, Set Background Color button
CLI :	calibration_BackgroundColor <i>ColorValue</i> <i>ColorValue: Red 0-255, Green 0-255, Blue 0-255</i>
SDK:	-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");	

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



17.9.14 Specify Calibration Stimulus Point Presentation Order	
GUI:	-none-
CLI :	calibration_CustomOrderList n1 n2 n3 n4 n5 n6 ...
SDK:	-none-
Default:	6 5 4 3 2 1 9 8 7 12 11 10 16 15 14 13 20 19 18 17 21 22 23 ...
<p>Allow specification of up to 72 calibration point in the desired order of presentation. <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">calibration_SelectIndexcalibration_SelectPointcalibrationPoints	
VPX_SendCommand ("calibration_ CustomOrderList 6 1 5 2 4 3");	

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



17.9.16	Display Custom Calibration Stimulus Point Order
GUI:	-none-
CLI :	calibration_CustomOrderDump
SDK:	-none-
Default:	
Prints the customOrder of calibration points to the History window.	
VPX_SendCommand ("calibration_CustomOrderDump");	

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



17.9.18 Select the Index Number that Maps to the Specified Calibration Data Point	
GUI:	-none
CLI :	<code>calibration_SelectIndex <i>indexSelection</i></code> <i>indexSelection: LAST, NEXT, or int { 1 .. N }</i>
SDK:	-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>	
<code>VPX_SendCommand ("calibration_SelectIndex 7"); // look up Pt# in CustomList</code>	

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



17.9.20 Re-Present the Specified Calibration Data Point	
GUI:	EyeSpace window, Re-present button
CLI :	<code>calibrationRedoPoint CalibrationPoint</code> <i>CalibrationPoint</i> : Numbered point to select.
SDK:	<code>VPX_ReCalibratePoint (int SelectPoint);</code>
<p>Re-presents the specified calibration data point. The behavior is determined by the <code>calibration_SnapMode</code> and <code>calibration_AutoIncrement</code> specifications. The <i>CalibrationPoint</i> parameter is optional, if omitted, the currently selected calibration point will be re-presented. <i>Note</i>: the point number corresponds to the slider values in the EyeSpace window and does not depend on the calibration method chosen. <i>See also</i>: <code>calibration_SnapMode</code> <code>calibration_AutoIncrement</code></p>	
<code>VPX_SendCommand ("calibrationRedoPoint 7");</code> <code>VPX_SendCommand ("calibrationRedoPoint"); // defaults to current point</code>	

17.9.21 Specify Custom Calibration Stimulus Point Locations	
GUI:	-none-
CLI :	<code>calibration_CustomPoint 2 0.95 0.95</code>
SDK:	<code>VPX_SendCommand("calibration_CustomPoint %d %g %g", pointIndex, loc.x, loc.y);</code>
<p>Specifies custom locations of the calibration stimulus points. <i>Note</i>: 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. <i>See also</i>: <code>calibration_CustomPointsUsed</code> <code>calibration_CustomPointDump</code></p>	
<code>VPX_SendCommand ("calibration_CustomPoint 2 0.95 0.95");</code>	



17.9.22 Turn Custom Calibration Stimulus Point Location ON / OFF	
GUI:	-none-
CLI :	calibration_CustomPointsUsed <i>BoolValue</i> BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	VPX_SendCommand("calibration_CustomPointsUsed Yes");
Default:	No
Turns custom calibration stimulus point location on or off. See also: calibration_CustomPoint calibration_CustomPointDump	
calibration_CustomPointsUsed YES	

17.9.23 Print Locations of custom calibration stimulus points in EventHistory window	
GUI:	-none-
CLI :	calibration_CustomPointDump
SDK:	VPX_SendCommand("calibration_CustomPointDump");
Prints the custom calibration point list in the History window.. See also: calibration_CustomPoint calibration_CustomPointsUsed	
calibration_CustomPointDump	

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



17.9.25 Compensate for Slip	
GUI:	EyeSpace window, Slip-Correction button
CLI :	<code>calibrationSlipCorrection</code> <i>CalibrationPoint</i> <i>CalibrationPoint</i> : Numbered point to select.
SDK:	<code>VPX_SlipCorrection (int SelectPoint); // deprecated</code>
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. See also: calibration_SnapMode	
<code>VPX_SendCommand ("calibrationSlipCorrection 7");</code>	

17.9.26 Adjust Calibration Area	
GUI:	Controls window, Regions Tab, Calibration Region radio button
CLI :	<code>calibrationRealRect</code> <i>L T R B</i> <i>L T R B</i> : Normalized floating point values for the corners.
SDK:	-none-
Default:	<code>0.1 0.1 0.9 0.9</code>
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.	
<code>VPX_SendCommand ("calibrationRealRect 0.2 0.2 0.8 0.8");</code>	

17.9.27 Save Image of Eye at each Calibration Data Point	
GUI:	-none-
CLI :	<code>calibration_SaveEyeImages</code> <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-none-
Default:	Off
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.	
<code>VPX_SendCommand ("calibration_SaveEyeImages yes");</code>	



17.9.28	Specify Amount of Parallax Correction
GUI:	Controls window, Regions Tab, Parallax Correction radio button
CLI :	<code>parallaxCorrection_Slope</code> <i>float</i> <i>in range 0.5 – 1.5</i>
SDK:	-none-
Default:	0.82
This command sets the amount of parallax correction. Only applicable to the binocular SceneCamera systems. Refer to 7.8	
<code>VPX_SendCommand ("parallaxCorrection_Slope <i>float</i>");</code>	

17.9.29 Set custom calibration stimulus point based on gaze (scene) video content	
GUI:	-none-
CLI :	calibration_OnContent <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-none-
Default:	Off
<p>This calibration method allows the user to specify the position of custom calibration stimulus points at locations in the GazeSpace window based on the video content. (e.g. a finger, a cardboard plaque with calibration dots painted on it etc.) When in this mode, a right mouse click in the Gaze space window does two things:</p> <p>(a) sets custom calibration stimulus point at point clicked in GazeSpace, on scene video content, (b) sets calibration data point in snap (remote) fashion.</p> <p>Enables the user to set the locations of the CustomCalibrationPoints (calibration stimulus points) based on user right-mouse clicks in the SceneCamera window (GazeSpace window). Presumably this would be a mouse click upon certain video content, such as a finger tip when the user says "look at my finger tip".</p> <p>This is primarily important because it allows post-hoc calibration on pre-recorded movies.</p> <p>Remember that the calibration points are in column-major order, which means that successive points proceed top-to-bottom down one column after another. For example, if there are 12 calibration points, the entry of OnContent points should proceed down the four columns, starting at column #1, on the left, 1:top-leftEdge, 2:middle-leftEdge, 3:bottom-leftEdge; then down the next two center columns, column #2 and column #3, and finally down column #4, on the right.</p> <p>Setting calibration_OnContent to ON also sets the following:</p> <p>(a) CLI: calibration_CustomPointsUsed to ON (b) Advanced Calibration > Snap Calibration Mode * to ON (c) Advanced Calibration > Auto-Increment ++ to ON (d) Advanced Calibration > Presentation Order to SEQUENTIAL (e) Control > Display > Calib Region to ON (f) Control > Display > ROI Regions to OFF</p> <p>Setting calibration_OnContent to OFF sets the above booleans to the opposite values, but does not reset the Calibration Present Order</p>	
VPX_SendCommand ("calibration_OnContent yes");	

17.10 Controls: Criteria Controls

17.10.1	Specify amount of Smoothing
GUI:	Controls window, Criteria tab, Smoothing Points slider
CLI :	smoothingPoints <i>IntValue</i>
SDK:	VPX_SetSmoothing (<i>IntValue</i>);
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:</p> <p>smoothingMethod, velocityCriterion</p> <p>Modified: 2.8.2.44</p>	
VPX_SendCommand ("smoothingPoints 3");	

17.10.2	Specify Smoothing Algorithm to Apply
Control or Menu:	-none-
CLI :	smoothingMethod <i>method</i> <i>method</i> : SMA, EMA
SDK:	-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 CLI smoothingPoints N.</p> <p><i>Note</i>: The successfully set method is currently reported in the Event History window.</p> <p>See also:</p> <p>smoothingPoints</p> <p>Added: 2.8.2.44</p>	
VPX_SendCommand ("smoothingMethod EMA");	



17.10.3 Specify Velocity Threshold	
Control or Menu:	Controls window, Criteria tab, Saccade Velocity slider
CLI :	<code>velocityCriterion <i>NormalizedValue</i></code> <code>velocityThreshold <i>NormalizedValue</i></code> <i>NormalizedValue</i> : floating point number in range 0.0 to 1.0
SDK:	-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 total velocity plot.</p> <p>Note that the velocity magnitude, and consequently the required threshold, will be affected by the amount of smoothing applied.</p> <p>See also:</p> <ul style="list-style-type: none"><code>smoothingPoints</code><code>driftCriterion</code>	
VPX_SendCommand ("velocityCriterion 0.8"); // Preferred VPX_SendCommand ("velocityThreshold 0.45"); // OlderForm	

17.10.4 Specify amount of Drift Allowed	
Control or Menu:	Controls window, Criteria tab, Fixation Drift Allowed slider
CLI :	<code>driftCriterion <i>NormalizedValue</i></code> <i>NormalizedValue</i> : floating point number in range 0.0 to 1.0
SDK:	-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>See also:</p> <ul style="list-style-type: none"><code>velocityCriterion</code><code>penPlot +DRIFT</code> <p>Changes:</p> <p>Default value changed in version 2.8.3 from 0.1 to 0.03</p>	
VPX_SendCommand ("driftCriterion 0.025");	



17.10.5 Specify Pupil Aspect Ratio Failure Criterion	
GUI:	Controls window, Criteria tab, Pupil Aspect Criterion
CLI :	<code>pupilAspectCriterion</code> <i>NormalizedValue</i> <i>NormalizedValue</i> : a floating point number in range 0.0 to 1.0
SDK:	<code>VPX_SetPupilOvalCriteria (<i>NormalizedValue</i>);</code>
Default:	0.05
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.	
<code>VPX_SendCommand ("pupilAspectCriterion 0.8");</code>	

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

17.11 Region of Interest (ROI)

17.11.1 Define an ROI Box	
GUI:	-none-
CLI :	<code>setROI_RealRect</code> <i>Index L T R B</i> <i>Index: Integer value indicating the ROI to specify.</i> <i>L T R B: Normalized floating point values for the corners.</i>
SDK:	<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> <ul style="list-style-type: none"> <code>setROI_AllOff</code> <code>setROI_isoEccentric</code> <code>VPX_GetROI_RealRect(n,rr);</code> <code>VPX_drawROI(HWND hWnd, int activeRegion);</code> 	
<pre>VPX_SendCommand ("setROI_RealRect 5 0.1 0.1 0.9 0.9"); // Sets ROI #5 with 10% margins</pre>	

17.11.2 Specify Number of ROI to be drawn in a circle around center of window	
GUI:	-none-
CLI :	<code>setROI_isoEccentric</code> <i>NumberOfBoxes</i> <i>NumberOfBoxes: integer number of ROI in a circle.</i>
SDK:	<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>	



17.11.3	Remove all ROI Boxes
GUI:	Controls window, DataDisplay tab, ROI Regions checkbox
CLI :	setROI_AllOff
SDK:	-none-
Removes all ROI boxes.	
VPX_SendCommand("setROI_AllOff");	

17.11.4	Select a Specific ROI
GUI:	Controls window, Regions tab, Region: slider
CLI :	setROI_Selection <i>ROIbox</i> <i>ROIbox</i> : 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:	-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"	
VPX_SendCommand("setROI_Selection 9");	

17.11.5	Select the next ROI Box
GUI:	Controls window, Regions tab, Region: slider F8
CLI :	setROI_selectNext
SDK:	-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.	
VPX_SendCommand("setROI_selectNext");	



17.11.6 Associate ROI with a particular stimulus image		
GUI:	menu: File > Image > Save Image ROI	Alt-Shift-I
CLI :	<code>savelImageROI</code> <code>savelImageROI differentFileName</code>	
SDK:	-none-	
<p>An easy way to associate an image and the regions of interest (ROI) for that image. Every time an image is loaded, e.g. e.g. <code>~/Images/MyImage01.bmp</code>, <i>ViewPoint</i> also looks for the file <code>~/Images/MyImage01.bmp_ROI.txt</code> that contains ROI specifications. These ImageROI files are really just special small Settings files that are closely associated with the image file. Though designed for ROI, they could also be edited, for example, to play a sound that is associated with the image.</p>		
<code>VPX_SendCommand("savelImageROI");</code>		

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

17.12 PenPlot controls

17.12.1 Specify Which PenPlot Traces to Display	
GUI:	Menu : PenPlots >
CLI :	<p>penPlot <i>+chartItemName - chartItemName</i></p> <p><i>chartItemName</i> : any of the following:</p> <p>All, Tming, Vergence, Xvelocity, Yvelocity, Tvelocity, Width, Aspect, Xgaze, Ygaze, Xangle, Yangle, Torsion, Seconds, HeadPosition, HeadAngle, Xpupil, Ypupil, Xglint, Yglint, Drift, Events, fixationTime, Timing, ROI, Quality, Processing, CPUload</p>
SDK:	-none-
Default:	+Xgaze +Ygaze +Velocity +Aspect +Drift
<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> <p>+Velocity +Width +Aspect +Timing</p> <p>+Vergence (data available with binocular option only)</p> <p>+HeadPosition (data available with head track option only)</p> <p>+HeadAngle (data available with head track option only)</p> <p>+xPupilPoint +yPupilPoint (raw eyespace pupil position data as a function of time)</p> <p>+XAngle +YAngle (data valid only when accurate GeometryGrid measurements are set)</p> <p>+Processing (the difference between the videoTime and the storeTime)</p> <pre>VPX_GetDataTime2(eyn, &videoTime); VPX_GetStoreTime2(eyn, &storeTime); processingTime = (storeTime - videoTime) * SecondsToMilliseconds;</pre> <p>+cpuLoad displays CPU load. Similar to the Windows Task Manager</p> <p>Note: you can specify the amount of smoothing to use for the display with the following command:</p> <pre>cpuLoad_SetSmoothing percentPreviousValue</pre> <p>Sets the amount of smoothing, must be ≥ 0.0 and < 1.0</p> <p>Lower values (e.g. 0.1) are less smoothing, higher values (e.g. 0.9) are more smoothing.</p> <p>The CPU usage varies wildly from sample to sample, so pretty much smoothing is required.</p> <p>Default: 0.7</p>	
<pre>VPX_SendCommand("penPlot +Xgaze +Ygaze -Tvelocity +Width +Aspect -Timing");</pre>	



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

17.12.3 PenPlot Limen Back Ground Color	
GUI:	-none-
CLI :	penPlot_LimenFillColor <i>ColorValue</i> <i>ColorValue: Red 0-255, Green 0-255, Blue 0-255</i>
SDK:	-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: penPlot_BackgroundColor	
VPX_SendCommand("penplot_LimenFillColor 115 100 235");	

17.12.4 Specify Speed of PenPlot Scrolling	
GUI:	Menu : PenPlots > Speed > { ¼ x, ½ x, 1x, 2x, 4x, 8x, 16x }
CLI :	penPlot_Speed <i>xInc</i> <i>xInc: float in range {0.25 to 16.0}</i>
SDK:	-none-
Default:	1
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, 2.8.3.445	
VPX_SendCommand("penPlot_speed 6");	



17.12.5 Specify Size of PenPlot Lines	
GUI:	Menu : PenPlots > PenSize > { 1, 2, 3, 4 }
CLI :	penPlot_PenSize <i>size</i> <i>size</i> : integer in range {1 to 4}
SDK:	-none-
Default:	1
Specifies the thickness of the pen plot trace lines. Modified: 2.8.3.445	
VPX_SendCommand("penPlot_penSize 4");	

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

17.12.7 Specify the behavior of the penpot after a video freeze	
GUI:	-none-
CLI :	penPlot_restartAfterFreeze <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-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.	
VPX_SendCommand("penplot_restartafterfreeze Yes");	

17.13 Graphics controls

17.13.1 Specify the color of the GazeSpace and PenPlot Lines	
GUI:	Controls window, Display tab, Eye A button or , Eye B button
CLI :	penColorA <i>ColorValue</i> penColorB <i>ColorValue</i> <i>ColorValue</i> : Red 0-255, Green 0-255, Blue 0-255
SDK:	-none-
Default:	penColorA: 100 255 0 (limeGreen); penColorB : 0 255 255 (cyan)
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.	
VPX_SendCommand("pencolorA 115 100 235"); VPX_SendCommand("pencolorB 0 255 0"); // Green	
17.13.2 Specify which Overlay Graphics to Display in the GazeSpace Window	
GUI:	Controls window, Display tab, Gaze option checkbox matrix
CLI :	gazeGraphicsOptions <i>+graphicsOptions -graphicsOptions</i> <i>graphicsOptions</i> : +ROI +POG +Path +Fix +Size +Grid +Cal +Raw +Time
SDK:	-none-
Default:	+ROI +Path +Image
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 word, i.e., NOT separated from it by a space. Separate multiple arguments by spaces, as in the example here below. <i>See also: stimulusGraphicsOptions</i>	
VPX_SendCommand("gazeGraphicsOptions +POG -Raw -ROI");	



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

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



17.13.5 Automatically erase display windows	
GUI:	-none-
CLI :	<code>timedErase_autoErase</code> <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-none-
Default:	Off
Automatically erase display windows. See also: timedErase_delaySeconds eraseDisplays	
VPX_SendCommand("timedErase_autoErase On"); VPX_SendCommand("timedErase_ delaySeconds 6.5");	

17.13.6 Specify time delay for auto erase	
GUI:	-none-
CLI :	<code>timedErase_delaySeconds</code> <i>seconds</i> <i>seconds</i> : floating point number
SDK:	-none-
Default:	Off
Automatically erase display windows. See also: timedErase_autoErase eraseDisplays	
VPX_SendCommand("timedErase_autoErase On"); VPX_SendCommand("timedErase_ delaySeconds 6.5");	

17.14 Stimulus Window controls

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



17.14.2 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
CLI :	<code>stimWind_CustomStatic L T R B</code> <i>L T R B: Normalized floating point values for the physical dimensions.</i>
SDK:	<code>VPX_StimWind_CustomStatic ("int x1, int y1, int x2, int y2");</code>
<p>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.</p> <p>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.</p> <p><i>Note:</i> obsolete term 'customStimulusWindow' would raise the window.</p> <p>See also:</p> <ul style="list-style-type: none"><code>stimWind_FullDisplay Custom</code><code>setWindow Stimulus Show</code><code>stimulus_ImageHidden</code>	
<code>VPX_SendCommand("stimWind_CustomStatic 1280 0 2304 768");</code> <code>VPX_SendCommand("setWindow Stimulus Show");</code>	



17.14.3	Automatically Show the Stimulus Window on Primary Monitor
GUI:	Stimuli> Stimulus Window Properties > AutoShow on Calibrate
CLI :	stimwind_AutoShowOnCalibrate <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	VPX_StimWind_AutoShowOnCalibrate (bool tf);
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>stimWind_FullDisplay 1</p>	
VPX_SendCommand("stimwind_autoshowoncalibrate No");	



17.14.4 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.
CLI :	stimWind_FullDisplay <i>OptionString</i> <i>MonitorNumber</i> : integer <i>OptionString</i> : Adjustable, Custom, 1, Primary, 2, Secondary
SDK:	VPX_StimWind_FullDisplay (int monitor) VPX_StimWind_Adjustable(void)
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> setWindow STIMULUS SHOW setWindow STIMULUS HIDE stimWind_CustomStatic stimWind_AutoShowOnCalibrate</p> <p><i>Modified: 2.8.2.52</i></p>	
VPX_SendCommand("stimWind_FullDisplay Secondary");	



17.14.5 Calibrate to a third party application window	
GUI:	-none-
CLI :	-none-
SDK:	VPX_SetExternalStimulusWindow(HWND hMyWnd);
<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 <i>same code in</i>:</p> <p style="padding-left: 40px;">VPX_MFC_Demo.cpp</p> <p><i>Contrast to:</i></p> <p style="padding-left: 40px;">HWND VPX_GetViewPointStimulusWindow();</p>	
<pre>CWnd* pWnd = GetDlgItem(IDC_StimulusPicture) ; HWND hWnd = pWnd->GetSafeHwnd() ; VPX_SetExternalStimulusWindow(hWnd);</pre>	

17.15 Window related controls

17.15.1 Print ViewPoint windows	
GUI:	File > Print > * window
CLI :	printWindow <i>windowNameString</i> <i>windowNameString</i> : Main, EyeCamera, EyeSpace, Controls, Status, GazeSpace, PenPlot
SDK:	-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 <i>also</i>:</p> <p style="padding-left: 40px;">printDateTimeStamp</p>	
<pre>VPX_SendCommand("printWindow EyeCamera");</pre>	



17.15.2 Include Date and Time Stamp on Printed windows	
GUI:	File > Print > Date Timestamp printouts (toggle)
CLI :	<code>printDateTimeStamp</code> BoolValue BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-none-
Default:	On
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. See: printWindow	
VPX_SendCommand("printDateTimeStamp Off");	

17.15.3 Size Window	
GUI:	Resize window with mouse.
CLI :	<code>sizeWindow</code> windowNameString W H windowNameString: Main, EyeCamera, EyeA, EyeB, EyeSpace, Controls, Status, GazeSpace, PenPlot, History, Events, Geogmetry, etc. W H: two integers describing the width and height of the window content rectangle.
SDK:	-none-
Sizes / resizes the specified window to the specified width and height. The left-top corner position remains the same; the right-bottom corner is adjusted to satisfy the new size. See also: setWindow moveWindow	
VPX_SendCommand("sizeWindow GazeSpace 640 480");	



17.15.4 Move Window, or Move & Resize Window	
GUI:	Drag window with mouse, resize window with mouse.
CLI :	<code>moveWindow <i>windowNameString</i> <i>L T</i></code> <code>moveWindow <i>windowNameString</i> <i>L T W H</i></code> <i>windowNameString</i> : Main, EyeCamera, EyeA, EyeB, EyeSpace, Controls, Status, GazeSpace, PenPlot, History, Events, Geogmetry, etc. <i>L T W H</i> : the first two integers specify the left, top corner of the window; the second two integers are optional and specify the width and height of the content area of the window.
SDK:	-none-
<p>Moves the specified window so the left-top corner is at the specified location. Optionally the window can be resized at the same time by additionally specifying the width and height.</p> <p>If the window is Free (see <code>setWindow Free</code>), the location can be outside the Viewpoint parent window, and can be on other monitors if multiple monitors are present.</p> <p>See also:</p> <ul style="list-style-type: none"><code>setWindow</code><code>sizeWindow</code>	
<code>VPX_SendCommand("setWindow GazeSpace Free; moveWindow GazeSpace -900 50");</code>	



17.15.5 Specify ViewPoint Window State, or Windows Layout	
GUI:	Windows > { WindowName } (Toggle) Windows > Arrange > { Startup Layout, Cascade } Window minimize and maximize title bar boxes.
CLI :	setWindow <i>windowNameString</i> <i>windowState</i> setWindow <i>windowLayout</i> <i>windowNameString</i> : Main, EyeCamera, EyeA, EyeB, EyeSpace, Controls, Status, GazeSpace, PenPlot, History, Events, Geogmetry, etc. <i>windowState</i> : Show, Hide, Maximize, Minimize, Restore, Free, Child <i>windowLayout</i> : Startup, Cascade
SDK:	int VPX_Video_WindowVisible (BOOL tf); int VPX_StimWind_Hide(void);
<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>Use the Free option to allow the specified window to be moved outside the <i>ViewPoint</i> parent window; use the Child option to make place it back inside the <i>ViewPoint</i> window.</p> <p>The Maximize option make the window full screen, and the Minimize option hides the window except for an icon at the bottom of the screen, just as the buttons in the window title bar do; the Restore option returns the window to the previous state.</p> <p><i>See also</i>:</p> <ul style="list-style-type: none">moveWindowsizeWindowstimWind_FullDisplaystimWind_AutoShowOnCalibrate <p><i>Deprecated</i>:</p> <ul style="list-style-type: none">stimWind_Hide	
VPX_SendCommand(“setWindow Stimulus Show”); VPX_SendCommand(“setWindow GazeSpace Free”); VPX_SendCommand(“setWindow PenPlot Maximize”);	



17.15.6 History options	
GUI:	-none-
CLI :	historyOptions +/-optionStrings optionStrings: +timeStamp
SDK:	-none-
Allows greater control of the information that appears in the History window. Use -graphicsOption to exclude an option, or use +graphicsOption to include an option. Note: The +/- must be next to the key word, i.e., NOT separated from it by a space.	
VPX_SendCommand("historyOptions -timeStamp"); // remove timestamp from history.	

17.15.7 Clear Event History window	
GUI:	Windows > Clear History
CLI :	eventHistory_Clear No arguments.
SDK:	-none-
Clears the EventHistory window.	
VPX_SendCommand("eventHistory_Clear");	

17.15.8 Save window layout settings	
GUI:	File > Settings > Save Window Layout
CLI :	settingsFile_SaveWindowLayout filename filename: Name of the settings file to be saved.
SDK:	-none-
The window layout information can be saved in a Settings file. It is not saved as part of the standard Save Settings operation. The file is automatically saved with a .txt extension. See also: settingsFile_Save settingsFile_Load	
VPX_SendCommand("settingsFile_SaveWindowLayout windowlayout_large");	

17.16 Settings File commands

17.16.1 Load Settings File	
GUI:	File > Settings > Load Settings ... ^L
CLI :	settingsFile_Load filename filename: Name of the settings file to be loaded.
SDK:	-none-
Loads a settings file of the specified file name. Recursion is not allowed. Nesting depth is limited to 9. <i>See also:</i> settingsFile_SaveWindowLayout	
VPX_SendCommand("settingsFile_Load researchsettings");	
17.16.2 Edit Settings File	
GUI:	File > Settings > Edit Settings File ...
CLI :	settingsFile_EditDialog
SDK:	-none-
Enables user to quickly access and edit Settings files.	
VPX_SendCommand("settingsFile_EditDialog");	
17.16.3 Show Verbose Settings File Loading Details in Event History	
GUI:	File > Settings > Verbose Loading
CLI :	settingsFile_Verbose BoolValue BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-none-
Default:	No
Specifies whether to report verbose loading details.	
VPX_SendCommand("setingsFile_verbose Yes");	



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

17.17 SettingsFileList commands

17.17.1 Initialize Settings File List		
GUI:	-none-	
CLI :	settingsFileList_Init	
SDK:	-none-	
Initializes the list for settings file, making it ready for new settings to be entered.		
VPX_SendCommand("settingsFileList_Init");		

17.17.2 Next Settings File in List		
GUI:	File > settings > SettingsFileList > Next Settings File	F9 (default)
CLI :	settingsFileList_Next	
SDK:	-none-	
Executes the settings for the next settings file in the settings file list.		
VPX_SendCommand("settingsFileList_Next");		

17.17.3 Add Settings File to the List		
GUI:	-none-	
CLI :	settingsFileList_AddName FileName FileName: Name of the settings file to be added to the settings file list.	
SDK:	-none-	
Adds a settings file name to the settings file list.		
VPX_SendCommand ("settingsFileList_AddName subject1.txt");		



17.17.4 Restart Settings File List	
GUI:	File > Settings > SettingsFileList > Restart SettingsFileList
CLI :	settingsFileList_Restart
SDK:	-none-
Reopens the settings file list and re-applies the settings listed in the setting files.	
VPX_SendComand("settingsFileList_Restart");	

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

17.17.6 Specify delay between Settings Files in List	
GUI:	-none-
CLI :	settingsFileList_SequenceSeconds <i>TimeValue</i> <i>TimeValue</i> : Amount of sequence seconds.
SDK:	-none-
Creates a delay between loading settings-files from the settings-file-list. Users may also specify sequence-seconds within the settings files themselves rather than in a settings-file-list. 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.	
VPX_SendCommand("settingsFileList_SequenceSeconds 4.5");	



17.17.7	Randomize Settings Files
GUI:	-none-
CLI :	settingsFileList_Randomize
SDK:	-none-
Randomizes settings files in the list.	
VPX_SendCommand("settingsFileList_Randomize");	

17.18 Torsion commands

17.18.1	Start / Stop Torsion Calculations
GUI:	Torsion window, Start / Stop button ^T
CLI :	torsion_Calculation <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-none-
Default:	off
Starts and stops the torsion calculations. As a side effect this also adds or removes the torsion stripChart from the penPlot window. <i>Note 1</i> : 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. <i>See also</i> : VPX_GetTorsion2	
VPX_SendCommand("torsion_Calculation On");	



17.18.2	Adjust Start Point of Torsion Sampling Arc
GUI:	Torsion window, Angle slider
CLI :	torsion_SampleAngle <i>FloatDegrees</i> <i>FloatDegrees</i> : floating point value 0.0 to 360.0 degrees
SDK:	-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>	
VPX_SendCommand("torsion_SampleAngle 166");	

17.18.3	Adjust Radius of Torsion Sampling Arc
GUI:	Torsion window, Radius slider
CLI :	torsion_SampleRadius <i>FloatValue</i> <i>FloatValue</i> : normalized floating point number 0.01 - 0.99
SDK:	-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>	
VPX_SendCommand("torsion_SampleRadius 0.75");	



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

17.18.5 Display Real-Time Torsion Data	
GUI:	Torsion window, Real-time graphics checkbox
CLI :	torsion_RealTimeGraphics <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-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.	
VPX_SendCommand("torsion_RealTimeGraphics On");	



17.18.6 Adjust Torsion Measurement Range	
GUI:	-none-
CLI :	<code>torsion_MeasureDegrees</code> <i>FloatDegrees</i>
SDK:	-none-
Default:	<code>+/- 9.0</code>
<p>Adjusts the torsion measurement range. Default is +/- 9 degrees. 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: <code>torsion_ResolutionDegrees</code>. <i>See also:</i> <code>torsion_ResolutionDegrees</code></p>	
<code>VPX_SendCommand("torsion_MeasureDegrees 9.0");</code>	

17.18.7 Adjust Torsion Measurement Resolution	
GUI:	-none-
CLI :	<code>torsion_ResolutionDegrees</code> <i>FloatDegrees</i> <i>FloatDegrees</i> : floating point value between: 0.20 to 360.0
SDK:	-none-
Default	<code>0.5</code>
<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 <code>Torsion_MeasureDegrees</code>. <i>See for further discussion:</i> <code>torsion_MeasureDegrees</code></p>	
<code>VPX_SendCommand("torsion_ResolutionDegrees 0.80");</code>	



17.18.8 Set Autocorrelation Template	
GUI:	Torsion window, Set Template button
CLI :	torsion_SetTemplate
SDK:	-none-
Sets a new autocorrelation template.	
VPX_SendCommand("torsion_SetTemplate");	

17.19 Interface settings commands

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

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



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

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

17.20 RemoteLink & SerialPort controls

17.20.1 Connect / Disconnect Serial Port	
GUI:	Interface > Serial Port > Connect
CLI :	<code>serialPortConnect</code> <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-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");	



17.20.2 Specify Serial Data to Send	
GUI:	Interface > Serial Port > Nothing, Stream, Test, Events, SinglePacket
CLI :	serialPortSend Nothing, Stream, Test, Events, SinglePacket
SDK:	-none-
Specifies what type of serial data to send.	
VPX_SendCommand("serialPortSend Events");	

17.20.3 Send Serial Port Ping	
GUI:	Interface > Serial Port > Send Ping alt + shift + P
CLI :	serialPortPing
SDK:	-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.	
VPX_SendCommand("serialPortPing");	

17.21 HeadTracking commands

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



17.21.2 Specify whether to use local or global origin	
GUI:	HeadTrack > Use Local Origin
CLI :	<code>headTracker_useLocalOrigin</code> <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-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>	

17.21.3 Reset Origin for the Head Sensor	
GUI:	HeadTrack > Reset Local Origin
CLI :	<code>headTracker_resetLocalOrigin</code>
SDK:	-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>	

17.21.4 Set Position and Angle Origins	
Control or Menu:	-none-
CLI :	<code>headTracker_setLocalOrigin</code> <i>coordinatesAndAngleValues</i> <i>coordinatesAndAngleValues</i> : x y z roll pitch yaw
SDK:	-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>	



17.21.5 Specify the Vector between Head Sensor and the Eyeball.	
Control or Menu:	-none-
CLI :	headTracker_eyeOffset <i>Coordinates</i> <i>Coordinates: x y z</i>
SDK:	-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.	
VPX_SendCommand("headTracker_eyeOffset 0.7 0.5 0.3");	

17.21.6 Turn CRT pulse synchronization On / Off	
GUI:	HeadTrack > CRT Sync > * Off, 50-72 Hz, 73-144 Hz
CLI :	headTracker_CRT_sync Off, Low, High
SDK:	-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.	
VPX_SendCommand("headTracker_CRT_sync Low");	

17.22 Binocular commands

17.22.1 Turn Binocular Mode On / Off	
GUI:	Binocular > Binocular mode (toggle)
CLI :	binocular_Mode <i>BoolValue</i> <i>BoolValue: Yes, No, True, False, On, Off, 1, 0, Toggle</i>
SDK:	-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.	
VPX_SendCommand("Binocular_Mode On");	

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

17.22.3 Specifies which eye to calibrate	
GUI:	EyeSpace window, Pull Down Menu > Eye A Eye B Both
CLI :	calibration_eyeTarget <arg> <arg> in { EyeA, EyeB, Both }
SDK:	-none-
Specifies whether to calibrate using Eye A, Eye B or both.	
VPX_SendCommand("calibration_eyeTarget Both ");	



17.23 File Related

17.23.1	Launch ViewPoint with Command Line Options
GUI:	-none-
CLI :	<code>launchApp <i>applicationName argumentsToApp</i></code> <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:	<code>VPX_LaunchApp(<i>applicationName, argsToApp</i>);</code>
<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 CLI strings are parsed by the ViewPoint application, therefore you cannot use this CLI string to launch the ViewPoint application itself, instead use: <code>VPX_LaunchApp("ViewPoint.exe", "");</code></p> <p>See also: 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>	

17.23.2	Quit ViewPoint
GUI:	File > Quit
CLI :	<code>quitViewPoint</code>
SDK:	<code>VPX_QuitViewPoint();</code>
<p>Terminates the ViewPoint EyeTracker application.</p> <p>See also: <code>VPX_LaunchApp</code></p>	
<code>VPX_SendCommand ("quitViewPoint");</code>	



17.23.3 Specify Default Folder paths	
GUI:	-none-
CLI :	<code>setPath pathID pathString</code> <i>pathID</i> : IMAGES: DATA: SETTINGS: SOUNDS: DOCUMENTATION: <i>pathString</i> : full path string or keyterm: DEFAULT_PATH
SDK:	-none-
<p>Specifies the current default folder path for important ViewPoint directories.</p> <p>The special keyTerm DEFAULT_PATH may be used to reset the path to the original ViewPoint default path string. Usually this is a folder directly under the main ViewPoint folder.</p> <p><i>Note</i>: The colon is part of the pathID and must be include, e.g.: IMAGES:</p> <p><i>Note</i>: paths may be set across machines, for example via mapped network drives, however saving data across such a path may cause delays or data loss.</p> <p>Version 2.8.3.44 improves behavior as follows:</p> <ul style="list-style-type: none">a) accepts both forward slashes and backward slashes, and convert them to forwards.b) appends a final slash at the end of the string, if it was not included by the user.c) tests that the path exists and report an error if it does not. <p><i>See also</i>:</p> <pre>PTCHAR VPX_GetViewPointHomeFolder(PTCHAR pathString);</pre>	
<code>setPath IMAGES: DEFAULT_PATH</code> <code>setPath IMAGES: "C:/ARI/Global/Images/"</code>	

17.24 FKey

17.24.1 Associate CLI s with FKeys	
GUI:	-none-
CLI :	<i>Fkey_cmd fKeyNumber commandString</i> <i>fKeyNumber</i> : integer in { 1 to 12 } <i>commandString</i> : any valid command
SDK:	-none-
<p>Allows CLI s to be associated with FKeys. It is a very useful to customize the Fkeys for your needs. 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>	

17.25 TTL

17.25.1 Associate CLI s with TTL Voltage Changes	
GUI:	-none-
CLI :	<i>ttl_cmd signedChannel commandString</i> <i>signedChannel</i> : +/- just before an integer in { 0 to 7 } <i>commandString</i> : any valid command
SDK:	-none-
<p>Allows CLI s to be associated with TTL voltage changes, high or low. 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>	



17.25.2 Set TTL Output Voltages	
GUI:	-none-
CLI :	<code>ttl_out <i>signedChannel</i></code> <i>signedChannel</i> : +/- just before an integer in { 0 to 7 }
SDK:	-none-
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.	
VPX_SendCommand("ttl_out -0"); // set TTL channel 0 LO VPX_SendCommand("ttl_out +7"); // set TTL channel 7 HI	

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

17.25.4 Print TTL values in the History Window	
GUI:	-none-
CLI :	<code>ttl_values</code>
SDK:	<code>int inValues = VPX_GetStatus(VPX_STATUS_TTL_InValues); // subject to change</code> <code>int outValues = VPX_GetStatus(VPX_STATUS_TTL_OutValues); // subject to change</code>
The CLI prints the TTL In and the TTL OUT values in the History window. The SDK interface returns an integer with the high channels bit coded as ones and the low channels bit coded as zeros. May not be available in the future through the VPX_GetStatus function.	
VPX_SendCommand("ttl_values");	



17.25.5 Set TTL Output to Indicate Data Quality Codes	
GUI:	-none-
CLI :	<code>ttl_out_quality channel levelString</code> <i>channel</i> : integer in { 0 to 7 } <i>levelString</i> : 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:	-none-
<p>A ttl output channel can be specified to indicate when the data quality value is greater than or equal to (>=) the quality critereon level.</p> <p>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 VPX_QUALITY_PupilScanFailed will cause the TTL channel to always be high, because the data quality value is always greater than or equal to this.</p> <p>See also:</p> <ul style="list-style-type: none">VPX_GetDataQualityverbose +ttl_out	
<code>VPX_SendCommand("ttl_out_quality 0 VPX_QUALITY_PupilFallBack");</code>	

17.26 Misc

17.26.1	Specify Verbose Information to Send to History Window
GUI:	-none-
CLI :	verbose +/- <i>activityType</i> <i>activityType</i> : ttl_out ttl_cmd calibration settings
SDK:	-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> <ul style="list-style-type: none"> +setting +parsing +ttl_out +ttl_cmd +frameGrabber +videoTiming +serialSend +serialReceive; +insertMark +insertString +insertUserTag +calibration +headTracker +nameList 	
<pre>VPX_SendCommand ("verbose +ttl_cmd +ttl_out -calibration -settings");</pre>	



17.26.2 Update Eye Data on Request	
GUI:	Alt-Shift-U
CLI :	updateData
SDK:	-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>	
VPX_SendCommand ("updateData");	

17.26.3 Set Status Window Update Rate for FPS Field	
GUI:	-none-
CLI :	fpsUpdate <i>nth_Interrupt</i> <i>nth_Interrupt</i> : integer
SDK:	-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>	
VPX_SendCommand ("fpsUpdate 3");	

17.26.4 SDK Debug Mode	
GUI:	-none-
CLI :	debugSDK <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	VPX_DebugSDK(int onOff)
Default:	Off
<p>This command will add debugging capability for the dll based sdk.</p>	
VPX_DebugSDK(1);	



17.26.5 Specify ViewPoint Generated Events	
GUI:	-none-
CLI :	vpx_event +option -option option : videoSynch
SDK:	-none-
<p>This provides a mechanism to control (thin) the number of events that <i>ViewPoint</i> generates. The command format the keyword <i>vpx_event</i> 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: VPX_VIDEO_SyncSignal</p>	
<code>VPX_SendCommand("vpx_event +videoSynch");</code>	

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



17.26.7	Confirm Quit
GUI:	-none-
CLI :	confirmQuit <i>BoolValue</i> <i>BoolValue</i> : Yes, No, True, False, On, Off, 1, 0, Toggle
SDK:	-none-
Default:	On
Specifies whether ViewPoint should ask to confirm a quit request, before terminating the program. It is suggested that this specification be placed in the StartUp.txt settings file. New: 2.8.4	
confirmQuit YES	

17.27 Parser Instructions

17.27.1	Settings File Comment
GUI:	-none-
CLI :	COMMENT //
SDK:	-none-
For use in a settings file. Use either the key word COMMENT or the double forward slashes (//) to tell the parser to ignore this line.	
//	

17.27.2	End of Settings File Command
GUI:	-none-
CLI :	END
SDK:	-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 18. Software Developers Kit (SDK)

18.1 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_SendCommand("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.

18.2 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.

18.3 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;
}
}
}

```

18.4 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_GlintsGood: 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.

18.5 Sending CLI s with the SDK

The *ViewPoint EyeTracker*® software developers kit (SDK) provides a routine that allows sending Command Line Interface (CLI) 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 CLI s, SDK functions, window buttons, and menu items, refer to the Command and Control Matrix contained in the SDK folder.

18.6 VPX_SendCommand(“setSomething”) replaces VPX_SetSomething

Previous versions of ViewPoint provided individual SDK functions to set contols (VPX_Set* commands), such that there were equivalent GUI, SDK, and CLI 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 CLI string that can be sent.**

18.7 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.

18.8 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();
```

18.9 SDK Trouble Shooting

This sections describes some of the common problems users encounter when programming with the SDK and how to solve them:

1. A constant stream of zero values obtained for eye data:
Make sure that you are linking to the same dll as *ViewPoint*.

18.10 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.



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

As of version 2.8.4.524, the major-axis and the minor axis are both normalized by the width of the EyeCamera, so the scales of the axes are commensurable.

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

The meaning of the values depends upon the Pupil Method that is selected: (a) for *Ellipse* dim.x and dims.y are the major and minor axes of the rotated ellipse, regardless of rotation angle, (b) for *OvalFit* dims.x is the horizontal width of the unrotated ellipse and dims.y is the vertical height, (c) for *Centroid* no pupil size is obtained.

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 circularity. This ratio value is independent of the EyeCamera window shape. A perfectly circular pupil will produce a value of 1.0. If the minor-axis is half that of the major axis, the value is 0.5. The minor-axis is always the numerator. So this circularity measure is always between 0.0 and 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, ¢roid);



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, °rees);

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);

18.10.2 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 synch 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 <i>dataTime</i> 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);

18.10.3 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 on the machine that the dll is running. Consequently, the return value will be different on the ViewPoint machine and a remote machine. See also: cursor_Control VPX_GetMeasuredScreenSize
<pre>//Convert from pixel position to normalized screen position. POINT pixelCursorPosition = VPX_GetCursorPosition(); VPX_RealPoint screenSize, normalizeCursorPosition, np; VPX_GetMeasuredScreenSize (&screenSize); // MUST be set correctly by user. normalizeCursorPosition.x = (float) pixelCursorPosition.x / screenSize.x ; normalizeCursorPosition.y = (float) pixelCursorPosition.y / screenSize.y ; //Put remoteMachine cursorPosition into ViewPoint dataFile. np = normalizeCursorPosition ; VPX_SendCommand("dataFile_insertString \"\t%g\t%g\" ", np.x, np.y);</pre>

18.10.4 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_VideosFrozen );
int open   = VPX_GetStatus( VPX_STATUS_DataFilesOpen );
int paused = VPX_GetStatus( VPX_STATUS_DataFilesPaused );
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"
```



18.10.5 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);
    }
}
```

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_GetMeasuredViewingDistance( &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);

18.10.6 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 <i>ViewPoint</i> 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 function 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_GetEventListItem](#)

[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_GetEventListItem (EyeType eye , int NthHit)

This only returns values when the gazePoint has just entered, or just exited an ROI. The gazePoint may have exited one or more ROI and also entered one or more ROI at the same data time, consequently we provide a list of ROI events. This function returns a positive ROI number to indicate entry into that ROI, and a negative value to indicate exit. The *NthEvent* argument starts at zero. The function may be called repeatedly to obtain successive ROI events until no more ROI are in the event list. If the *NthEvent* argument is greater than the event count, then the function returns the value ROI_NO_EVENT. The test is for the gazePoint entering inside the ROI, not resting on the boundary lines.

Monocular *ViewPoint* users should specify Eye_A.

See also: [VPX_ROI_GetHitListItem](#)

```
int ix = 0;
while( ROI_NO_EVENT != ( roiNumber = VPX_ROI_GetEventListItem( EYE_A, ix++ ))) if (roiNumber<0) doExiting(
roiNumber ) else doEntering( roiNumber );
```



```
int VPX_ROI_MakeHitListString (  
    EyeType eye ,  
    char *dataString,  
    int maxStringLength,  
    BOOL indicateOverflow,  
    char *noHitsString )
```

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](#)

```
char myString [80]="";  
VPX_ROI_GetHitListItem( EYE_A, myString, 80, true, "No ROI were hit" );
```

18.10.7 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:

```
vpv_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 );
```


18.11 DLL Interface

SDK API & Messaging Functions
<p>VPX_InsertMessageRequest (HWND hWnd, UINT msg)</p> <p>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</p>
<p>VPX_RemoveMessageRequest (HWND hWnd)</p> <p>Removes your application's request for notification for the specified window.</p>
<p>VPX_GetMessageListLength (int * num)</p> <p>Returns the number of windows that are registered to receive messages.</p>
<p>VPX_GetMessagePostCount (int * num)</p> <p>Returns the total number of messages that have been distributed.</p>
<p>int VPX_GetViewPointAppCount(int *apps);</p> <p>Sets applications to non-zero if ViewPoint is running.</p>
<p>BOOL VPX_VersionMismatch (VPX_SDK_VERSION)</p> <p>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.</p>
<p>double VPX_GetDLLVersion()</p> <p>You can obtain the version number of the loaded DLL using this function.</p> <p>Example:</p> <pre>double dllVersion = VPX_GetDLLVersion(); if (VPX_SDK_VERSION != dllVersion) doSomething();</pre>
<p>VPX_STATUS_DistributorAttached</p> <p>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.</p> <pre>#define VPX_Distributor_None 0 #define VPX_Distributor_IsViewPoint 1 #define VPX_Distributor_IsRemoteLink 2 #define VPX_DistributorType int</pre> <pre>VPX_DistributorType dllDataSource = VPX_GetStatus(VPX_STATUS_DistributorAttached);</pre> <p>// Note: VPX_STATUS_ViewPointIsRunning returns true if ViewPoint is running either directly or via RemoteLink.</p>



SDK Utility Functions
<p>VPX_GetPrecisionDeltaTime (double*, resetHoldTime) Retrieves the delta time between the holdTime and the current time.</p>
<p>BOOL VPX_IsPrecisionDeltaTimeAvailableQ() Use this command to determine if precision time is supported. Returns TRUE if the system supports precision time, otherwise returns false.</p>
<p>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.</p>
<p>VPX_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.</p>
<p>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.</p>
<p>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 }</p>
<p>VPX_drawROI (HWND hWnd, int activeRegion) Draws the activeRegion ROI in red and all of the other ROI in blue, within the specified window.</p>
<p>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</p>

ViewPoint Events & Notification Messages

18.11.1 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 VPX_EyeType eyn = (VPX_EyeType)LOWORD(wparam); VPX_RealRect gpt; VPX_GetGazePoint2(eyn, &gpt);
LPARAM	Do not use LPARAM.

HIWORD(WPARAM)	VPX_ROI_CHANGE Indicates that a Region Of Interest (ROI) was changed.
LOWORD(WPARAM)	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);
LPARAM	Do not use LPARAM.

HIWORD(WPARAM)	VPX_STATUS_CHANGE Indicates that a key ViewPoint status item was changed. For details, see Section: 18.10.4: Get ViewPoint Status. VPX_GetStatus
LOWORD(WPARAM)	Do not use WPARAM.
LPARAM	WORD statusItem = LOWORD(lParam); WORD statusValue = HIWORD(lParam); switch (statusItem) { case VPX_STATUS_DataFilesOpen : printf("DataFile is %s", (statusValue==1)?"Open":"Closed"); break; ... }



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!

18.11.2 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)	VPX_CAL_BEGIN Indicates that a calibration sequence is about to start. 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.
LOWORD(WPARAM)	The calibration point number, the actual point index number, not the random or custom sequence number.
LPARAM	The location of the upcoming stimulus point. <i>// Contains the location of the upcoming stimulus point. As below, use: VPX_LParam2RectPoint(lParam, cr, &calPt);</i>

HIWORD(WPARAM)	VPX_CAL_WARN Provides an opportunity to display a warning message to the subject, to make sure that they are paying attention. Follows VPX_CAL_BEGIN. 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.
LOWORD(WPARAM)	The calibration point number, the actual point index number, not the random or custom sequence number.
LPARAM	Contains the location of the upcoming stimulus point. <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>



HIWORD(WPARAM)	VPX_CAL_SHOW 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.
LOWORD(WPARAM)	The calibration point number, the actual point index number, not the random or custom sequence number.
LPARAM	Contains the location of the stimulus point. As above, use: VPX_LParam2RectPoint(IParam, cr, &calPt);

HIWORD(WPARAM)	VPX_CAL_ZOOM 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.
LOWORD(WPARAM)	The stimulus radius (shrinks from 15 to 2). WORD zoomSize = LOWORD(wParam);
LPARAM	Contains the location of the stimulus point. POINT pt ; WORD r, zoomSize = LOWORD(wParam); RECT cr ; GetClientRect(hwnd, &cr); VPX_LParam2RectPoint(IParam, cr, &pt); r = cr.right * zoomSize / 200 ; Rectangle (hdc, pt.x - r, pt.y - r, pt.x + r, pt.y + r);



HIWORD(WPARAM)	VPX_CAL_SNAP Indicates that the calibration image of the eye is being taken 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.
LOWORD(WPARAM)	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. <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	Contains the location of the stimulus point. As above, use: VPX_LParam2RectPoint(lParam, cr, &calPt);

HIWORD(WPARAM)	VPX_CAL_HIDE Indicates completion of the current calibration point. The program should clean up any remnants of this last calibration stimulus point display. Follows VPX_CAL_SNAP.
LOWORD(WPARAM)	The calibration point number, the actual point index number, not the random or custom sequence number. <pre>int pointNumber = LOWORD(wParam)</pre>
LPARAM	Contains the location of the stimulus point. As above, use: VPX_LParam2RectPoint(lParam, cr, &calPt);



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.</p> <p>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.</p> <p><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>BOOL doSlipFix = LOWORD(wParam) == 1 ;</p>
LPARAM	<p>Contains the location of the stimulus point.</p> <p>As above, use: VPX_LParam2RectPoint(lParam, cr, &calPt);</p>

18.12 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.

18.12.1 Old CLI

Old Name	New Name	Reason
circularPupilCriteria	pupilAspectCriterion	Singular / Clarity
calibrationSpeed	calibration_StimulusDuration	Clarity
gazeGraphics	gazeGraphicsOptions	Easier string parser
stimulusGraphics	stimulusGraphicsOptions	Easier string parser
pupilMinWidthCriteria	pupilMinWidthCriterion	Singular
pupilAspectCriteria	pupilAspectCriterion	Singular
circularPupilCriteria	pupilAspectCriterion	Clarity & Singular
pupilMaxWidthCriteria	pupilMaxWidthCriterion	Singular
cursorControl	cursor_Control	Consistency
gazeColor	penColorA	Clarity & binoc

18.12.2 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_GetDataQuality](#).

Old positional bit code was cumbersome.
New string parser is easier and clearer

Chapter 19. Troubleshooting

This section discusses some of the common sources of error and problem areas. Once recognized, many of these can be avoided.

19.1 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 CLI.

19.2 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.

19.3 EyeCameraWindow Troubleshooting

If a video source was connected to the computer when *ViewPoint™* was started, the **EyeCamera** window should display the captured video image. Otherwise, follow the following troubleshooting tips:

- the **EyeCamera** window shows “*** FROZEN ***”, then you should select the snowflake icon on the **EyeCamera** window to 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: **EyeCamera** window > monitor icon > reset EyeCamera

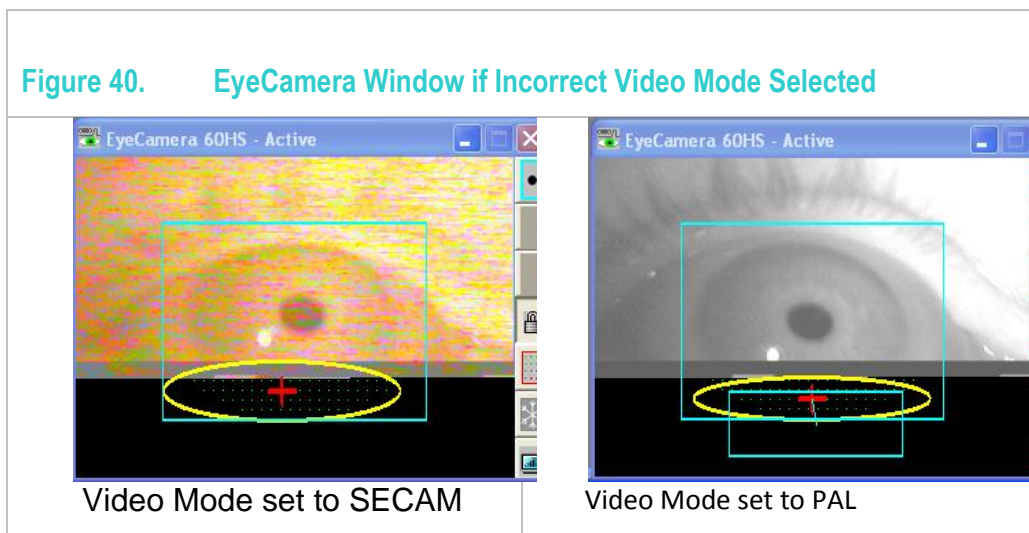
Note: If a camera is disconnected and reconnected then ViewPoint will automatically reset the video pipe and start working again. If after eight reset tries the software cannot detect a camera then the message MaxVideoResetTries exceeded will appear in the Status window.

- 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).

19.3.1 Bottom half of EyeCamera window is black

If the bottom of EyeCamera image is black as in Figure 40 Then the video standard has been set to PAL or SECAM, but the camera is NTSC. All ARI supplied cameras are NTSC. Select the monitor icon on the EyeCamera window >> Video Standard > NTSC. Also check any Settings file that may be loaded, e.g. StartUp.txt, that may specify a different video standard.



19.4 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 20. Error Codes

20.1 Introduction to Error Codes

Check whether the error code has a description and remedy below. If not, ViewPoint error codes provide a good way for customers to precisely communicate problems to our support team, and thus more quickly obtain useful help.

Error #	Error String	Remarks & Remedies
0	VP_ERROR_Undefined	This is a place holder and should be used.
	Launch errors	
1	VP_ERROR_UnsupportedOS	
2	VP_ERROR_MaxLicenses	
3	VP_ERROR_DllsLocked	
4	VP_ERROR_AppCountTwoExceeded	
5	VP_ERROR_AppCountOneExceeded	
22	VP_ERROR_ElevatedVistaModeRequired	Run app as administrator
120	VP_ERROR_ControlTabInit	
121	VP_ERROR_ControlTabNullWindow	
122	VP_ERROR_ControlTabInsertItem	
123	VP_ERROR_ControlTabsNull	
130	VP_ERROR_ListBoxDeleteStringError	
140	VP_ERROR_LoadMenuFailed	
150	VP_ERROR_normalValue_fromSlider	
220	VP_ERROR_StimulsDisplay	
221	VP_ERROR_HideStimulsWindowError	
	200s file errors	
201	VP_ERROR_BadDataFileHandle	
202	VP_ERROR_FileNameIllegalChars	
203	VP_ERROR_FileNameEmptyString	
204	VP_ERROR_ImageFileLoad	
205	VP_ERROR_EyeMovieSetThreadPriority	
	300s are PenPlot errors	
300	VP_ERROR_PenPlots	
301	VP_ERROR_PenPlot_HiLoMismatch	
302	VP_ERROR_PenPlot_InvalidEye	
303	VP_ERROR_PenPlot_OutOfRange	
	400s are Printer errors	
401	VP_ERROR_PrintInvalidHandle	
402	VP_ERROR_PrinterNotFound	
403	VP_ERROR_PrintRasterNotAllowed	
404	VP_ERROR_PrinterGetDefault	



405	VP_ERROR_PrinterGetDateStamp	
406	VP_ERROR_PrinterGetTimeStamp	
	500s are License & Decrypt errors	
502	VP_ERROR_CryptFileEndError	
503	VP_ERROR_CryptFileDataInvalid	May have unreadable characters in file
504	VP_ERROR_LicenseDataOpenError	
505	VP_ERROR_DecryptCreateKeyContainer	
506	VP_ERROR_DecryptServiceHandle	
507	VP_ERROR_DecryptHeaderLength	
508	VP_ERROR_DecryptMemoryAlloc	
509	VP_ERROR_DecryptFileHeader	
510	VP_ERROR_DecryptCryptImportKey	
511	VP_ERROR_CryptCreateHash	
512	VP_ERROR_CryptHashData	
513	VP_ERROR_CryptDeriveKey	Windows98 error, must use Windows XP
514	VP_ERROR_CryptDestroyHash	
515	VP_ERROR_DecryptOutOfMemory	
516	VP_ERROR_DecryptReadingCiphertext	
517	VP_ERROR_CryptDecrypt	
518	VP_ERROR_DecryptSourceClose	
519	VP_ERROR_CryptReleaseContext	
520	VP_ERROR_LicenseDirectoryMissing	
521	VP_ERROR_LicenseFileMissing	
522	VP_ERROR_LicenseFileInvalid	
523	VP_ERROR_LicenseNumberInvalid	
524	VP_ERROR_CryptDestroyKey	Was previously 518
	600s are EyeMovie	
601	VP_ERROR_MovieFileNotFound	
602	VP_ERROR_MovieFileReadError	
603	VP_ERROR_MovieFileBadFormat	
604	VP_ERROR_MovieFileMemoryError	
605	VP_ERROR_MovieFileOpenError	diagnosis of exclusion
606	VP_ERROR_EyeMovieSetThreadPriority	
607	VP_ERROR_EyeMovie_NoVideoStream	
	700s video errors	
701	VP_ERROR_VideoInitializationFailed	was previously 101
702	VP_ERROR_VideoStartupFailed	was previously 102
703	VP_ERROR_VideoChannelFailed	was previously 103
	900s ethernet errors	
901	VP_ERROR_EthernetError	Currently not used
	1000s are for bad function arguments	
1001	VP_ERROR_BadCalibrationSpeedValue	Must be in range: (1-400)
1002	VP_ERROR_BadCalibrationSIValue	Must be in range: (0-9)
1003	VP_ERROR_BadCalibrationWarningTimeValue	Must be in range: (1-100)
	2000s are for Date & Time	
2001	VP_ERROR_StringFormat_LongDate	
2002	VP_ERROR_StringFormat_ShortDate	
2003	VP_ERROR_StringFormat_TimeOfDay	
	3000s utilities	
3001	VP_ERROR_EnsureSubDirectory	
3002	VP_ERROR_EnsureSubDirectoryFolder	
3003	VP_ERROR_SettingsFolderCreatedNotice	



	4000s LC1 errors	
4001	VP_ERROR_LC1_InvalidVideoChannel	
4002	VP_ERROR_LC1_AllocImageBufferError	
4003	VP_ERROR_LC1_AddBufferToSequenceError	
4004	VP_ERROR_LC1_FreeBufferInSequenceError	
	9100s FOB HeadTracker	
9101	VP_ERROR_HT_WatchThread	
9102	VP_ERROR_HT_ConnectionFailed	
9103	VP_ERROR_HT_SerialPortSettings	
9104	VP_ERROR_HT_BirdSerialNumber	
9105	VP_ERROR_HT_SensorSerialNumber	
9106	VP_ERROR_HT_TransmitterSerialNumber	
9107	VP_ERROR_HT_CrystalSpeed	
	9200s RemoteLink SerialPort	
9201	VP_ERROR_RL_WatchThread	
9202	VP_ERROR_RL_ConnectionFailed	
9203	VP_ERROR_RL_SerialPortSettings	
9204	VP_ERROR_RL_Old27Data	
	9300s LC1 Capture Board errors	
9301	VP_ERROR_LC1_Not32BitColor	
9302	VP_ERROR_LC1_ExitBoard	
9303	VP_ERROR_LC1_SetThreadPriority	
9304	VP_ERROR_LC1_NoVideoChannels	
	9400s ViewPointMovie (VPM) errors	
9401	VP_ERROR_VPM_WriteFrame	
9402	VP_ERROR_VPM_OpenExisting	
9403	VP_ERROR_VPM_ReadFrame	
	9500 AVI 2.0 errors	
9500	VP_ERROR_AVI2_DxDIBService_CoCreate	
9501	VP_ERROR_AVI2_DxDIBService_Provider	
9502	VP_ERROR_AVI2_Writer_BadVideoChan	
9503	VP_ERROR_AVI2_Writer_ServiceFailed	
9504	VP_ERROR_AVI2_Writer_CodecListIndex	
9505	VP_ERROR_AVI2_Writer_CodecSetFailed	
9506	VP_ERROR_AVI2_Writer_RateSetFailed	
9507	VP_ERROR_AVI2_WriterCreateFileFail	
9508	VP_ERROR_AVI2_WriterFailedToAppend	
9509	VP_ERROR_AVI2_GetFrameByIndexFailed	
9510	VP_ERROR_AVI2_OkayToAccessQ_VidChan	
9511	VP_ERROR_AVI2_MissingLibsFolder	
	9600s Generic Movies	
9601	VP_ERROR_MovieTypeObsolete_AVI1	
9602	VP_ERROR_MovieFileSizelsZero	
	9700s COM, DCOM etc.	
9701	VP_ERROR_COM_ThreadModeChanged	
	9800s Parser, SettingsFile, CLI	
9801	VP_ERROR_CLI_ObsoleteCommand	

Chapter 21. 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.

21.1 Electrical Methods

21.1.1 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.

21.1.2 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.

21.2 Optical Methods

21.2.1 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.

21.2.2 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.

21.2.3 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.

21.2.4 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.

21.2.5 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.

21.2.6 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.



Arlington Research

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 22. Binocular Option

This section describes those features that are particular to the *ViewPoint EyeTracker*® binocular option only.

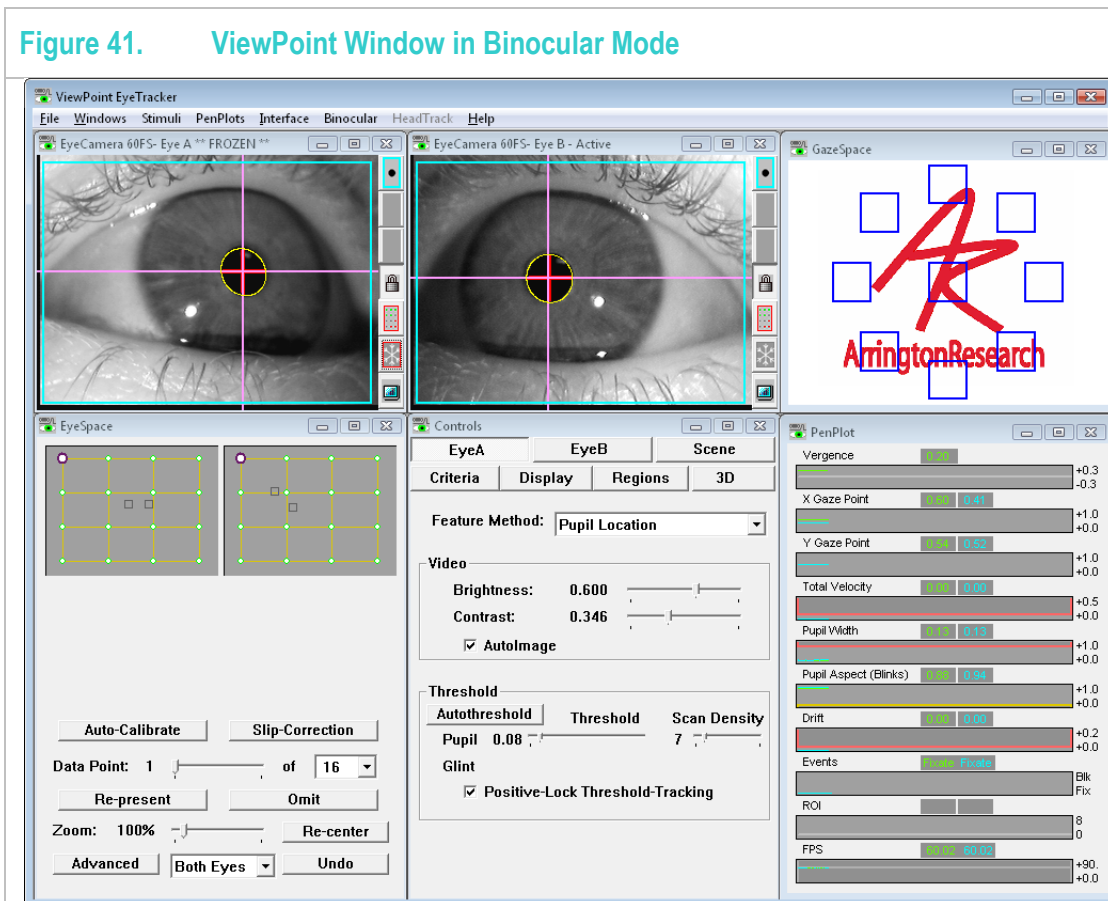
22.1 Installing Binocular FrameGrabber & Software

Refer to Chapter 4

22.2 Operating in Binocular Mode

To switch from monocular to binocular operation, select menu item: **Binocular > Binocular Mode** See Figure 41 below for the binocular mode window layout.

Figure 41. ViewPoint Window in Binocular Mode



Alternatively, use the Command Line Interface (CLI) 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 13

22.3 Setup

For optimal performance in the binocular mode, follow these suggestions:

- The video modes may be different between Eye-A and Eye-B if required.
- Set the video modes to High Precision rather than High Speed, unless the higher speed is actually required.
- When collecting data, avoid moving the mouse, especially to resize or move windows, which may cause video frame loss especially on slower computers.

When Binocular Mode is selected, the [GazeSpace](#) window has an additional drop down box to specify which eye the calibration process is to apply to. See Figure 41

22.4 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 11 for details of data file format.

22.5 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.

Note: The Binocular averaging option only affects the [GazeSpace](#) and [Stimulus](#) window displays.

Alternatively, use the Command Line Interface (CLI) 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 13

22.6 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. Use the newer Ethernet Interface. See Chapter 14

Chapter 23. 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



Arlington Research

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 your 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 environmental damage ("High Risk Activities"). ARI specifically disclaims any express or implied warranty of fitness for High Risk Activities.

© 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.

Chapter 24. 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.