POINT GREY

Triclops

Software Development Kit (SDK) Reference

Version 3.3 alpha 2
Revised June 24, 2010

Point Grey Research Inc.
12051 Riverside Way • Richmond, BC • Canada • V6W 1K7 • **T** (604) 242-9937 •
www.ptgrey.com

TRICLOPS SOFTWARE LICENSE AGREEMENT

The Triclops Stereo Vision Software Development Kit (the "Software") is owned and copyrighted by Point Grey Research, Inc. All rights are reserved. The Original Purchaser is granted a license to use the Software subject to the following restrictions and limitations.

1. The license is to the Original Purchaser only, and is nontransferable unless you have received written permission of Point Grey Research, Inc.
2. The Original Purchaser may use the Software only with Point Grey Research, Inc. cameras owned by the Original Purchaser, including but not limited to, Digiclops® and Bumblebee Camera Modules.
3. The Original Purchaser may make back-up copies of the Software for his or her own use only, subject to the use limitations of this license.
4. Subject to s.5 below, the Original Purchaser may not engage in, nor permit third parties to engage in, any of the following:
   A. Providing or disclosing the Software to third parties.
   B. Making alterations or copies of any kind of the Software (except as specifically permitted in s.3 above).
   C. Attempting to un-assemble, de-compile or reverse engineer the Software in any way.
   D. Granting sublicenses, leases or other rights in the Software to others.
5. Original Purchasers who are Original Equipment Manufacturers may make Derivative Products with the Software. Derivative Products are new software products developed, in whole or in part, using the Software and other Point Grey Research, Inc. products. Point Grey Research, Inc. hereby grants a license to Original Equipment Manufacturers to incorporate and distribute the libraries found in the Software with the Derivative Products. The components of any Derivative Product that contain the Software libraries may only be used with Point Grey Research, Inc. products, or images derived from such products.

   By the distribution of the Software libraries with Derivative Products, Original Purchasers agree to:
   A. not permit further redistribution of the Software libraries by end-user customers;
   B. include a valid copyright notice on any Derivative Product; and
   C. indemnify, hold harmless, and defend Point Grey Research, Inc. from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of any Derivative Product.

Point Grey Research, Inc. reserves the right to terminate this license if there are any violations of its terms or if there is a default committed by the Original Purchaser. Upon termination, for any reason, all copies of the Software must be immediately returned to Point Grey Research, Inc. and the Original Purchaser shall be liable to Point Grey Research, Inc. for any and all damages suffered as a result of the violation or default.

TABLE OF CONTENTS

1. INTRODUCTION 8

1.1. STEREO VISION TECHNOLOGY 8

2. INSTALLING THE SOFTWARE 9

2.1. INSTALLATION OF TRICLOPS 9
2.2. IMAGE VIEWING 10

3. GETTING STARTED 11

3.1. RUNNING THE DEMO PROGRAM 11
3.2. COMPILING A TRICLOPS PROGRAM 11

4. STEREO VISION DETAILS 12

4.1. ESTABLISHING CORRESPONDENCE 13
4.2. CALCULATING DISTANCES 13
4.3. TRICLOPS LIBRARY DATA FLOW 14
4.4. PREPROCESSING 15
4.5. RECTIFICATION 15
4.6. EDGE DETECTION 15
4.7. STEREO PROCESSING 16
4.8. DISPARITY RANGE 16
4.9. CORRELATION MASK 16
4.10. VALIDATION 16
4.11. BETTER CALIBRATION 17
4.12. SUBPIXEL INTERPOLATION 17
4.13. SURFACE VALIDATION 17
4.14. SUBPIXEL VALIDATION MAPPING 18
4.15. REGION OF INTEREST AND SUBPIXEL 18
4.16. DISPARITY MAPPING 18

5. PROGRAMMING WITH THE TRICLOPS APPLICATION PROGRAMMING INTERFACE (API) 19

5.1. PROGRAMMING WITH THE TRICLOPS STEREO VISION SDK 19
5.2. TRICLOPS CONTEXT 20
5.3. TRICLOPS EXAMPLE SOURCE CODE 20

6. TRICLOPS APPLICATION PROGRAMMING INTERFACE (API) FUNCTION REFERENCE 21

6.1. ENUMERATIONS 21
6.1.1. TriclopsCamera 21
6.1.2. TriclopsCameraConfiguration 22
6.1.3. TriclopsError 22
6.1.4. TriclopsImage16Type 24
6.1.5. TriclopsImageType 24
6.1.6. TriclopsInputType 25
6.2. TYPES 26

6.2.1. DisparityImageInfo ... *26*
6.2.2. EdgeImageInfo ... *26*
6.2.3. RectifiedImageInfo ... *27*
6.2.4. TriclopsBool ... *27*
6.2.5. TriclopsColorImage ... *27*
6.2.6. TriclopsContext ... *28*
6.2.7. TriclopsImage ... *28*
6.2.8. TriclopsImage16 ... *28*
6.2.9. TriclopsImage3d ... *29*
6.2.10. TriclopsImageInfo ... *29*
6.2.11. TriclopsInput ... *30*
6.2.12. TriclopsInputRGB ... *30*
6.2.13. TriclopsInputRGB32BitPacked ... *31*
6.2.14. TriclopsPackedColorImage ... *31*
6.2.15. TriclopsPackedColorPixel ... *32*
6.2.16. TriclopsPoint3d ... *32*
6.2.17. TriclopsRectImgQuality ... *32*
6.2.18. TriclopsROI ... *32*
6.2.19. TriclopsStereoQuality ... *33*
6.2.20. TriclopsTimestamp ... *33*
6.2.21. TriclopsTransform ... *34*
6.3. DEBUGGING AND ERROR REPORTING ... 34
6.3.1. triclopsErrorToString ... *34*
6.4. VALIDATION SUPPORT ... 34
6.4.1. triclopsGetBackForthValidation ... *34*
6.4.2. triclopsGetBackForthValidationMapping ... *35*
6.4.3. triclopsGetStrictSubpixelValidation ... *35*
6.4.4. triclopsGetSubpixelValidationMapping ... *35*
6.4.5. triclopsGetSurfaceValidation ... *36*
6.4.6. triclopsGetSurfaceValidationDifference ... *36*
6.4.7. triclopsGetSurfaceValidationMapping ... *37*
6.4.8. triclopsGetSurfaceValidationSize ... *37*
6.4.9. triclopsGetTextureValidation ... *37*
6.4.10. triclopsGetTextureValidationMapping ... *38*
6.4.11. triclopsGetTextureValidationThreshold ... *38*
6.4.12. triclopsGetUniquenessValidation ... *38*
6.4.13. triclopsGetUniquenessValidationMapping ... *39*
6.4.14. triclopsGetUniquenessValidationThreshold ... *39*
6.4.15. triclopsSetBackForthValidation ... *40*
6.4.16. triclopsSetBackForthValidationMapping ... *40*
6.4.17. triclopsSetStrictSubpixelValidation ... *41*
6.4.18. triclopsSetSubpixelValidationMapping ... *41*
6.4.19. triclopsSetSurfaceValidation ... *41*
6.4.20. triclopsSetSurfaceValidationDifference ... *42*
6.4.21. triclopsSetSurfaceValidationMapping ... *42*
6.4.22. triclopsSetSurfaceValidationSize ... *43*

6.4.23. triclopsSetTextureValidation ... *43*
6.4.24. triclopsSetTextureValidationMapping ... *43*
6.4.25. triclopsSetTextureValidationThreshold ... *44*
6.4.26. triclopsSetUniquenessValidation ... *44*
6.4.27. triclopsSetUniquenessValidationMapping ... *45*
6.4.28. triclopsSetUniquenessValidationThreshold ... *45*
6.5. GENERAL ... 46
6.5.1. triclopsBuildPackedTriclopsInput ... *46*
6.5.2. triclopsBuildRGBTriclopsInput ... *46*
6.5.3. triclopsGetImage ... *47*
6.5.4. triclopsGetImage16 ... *48*
6.5.5. triclopsSaveImage ... *48*
6.5.6. triclopsSaveImage16 ... *49*
6.5.7. triclopsVersion ... *49*
6.6. STEREO ... 50
6.6.1. triclopsGetDisparity ... *50*
6.6.2. triclopsGetDisparityMapping ... *50*
6.6.3. triclopsGetDisparityMappingOn ... *50*
6.6.4. triclopsGetDisparityOffset ... *51*
6.6.5. triclopsGetDoStereo ... *51*
6.6.6. triclopsGetEdgeCorrelation ... *52*
6.6.7. triclopsGetEdgeMask ... *52*
6.6.8. triclopsGetMaxThreadCount ... *52*
6.6.9. triclopsGetROIs ... *53*
6.6.10. triclopsGetStereoMask ... *53*
6.6.11. triclopsGetStereoQuality ... *54*
6.6.12. triclopsGetSubpixelInterpolation ... *54*
6.6.13. triclopsSetAnyStereoMask ... *54*
6.6.14. triclopsSetDisparity ... *55*
6.6.15. triclopsSetDisparityMapping ... *55*
6.6.16. triclopsSetDisparityMappingOn ... *56*
6.6.17. triclopsSetDoStereo ... *56*
6.6.18. triclopsSetEdgeCorrelation ... *57*
6.6.19. triclopsSetEdgeMask ... *57*
6.6.20. triclopsSetMaxThreadCount ... *57*
6.6.21. triclopsSetNumberOfROIs ... *58*
6.6.22. triclopsSetStereoMask ... *58*
6.6.23. triclopsSetStereoQuality ... *59*
6.6.24. triclopsSetSubpixelInterpolation ... *59*
6.6.25. triclopsStereo ... *60*
6.7. CONFIGURATION ... 60
6.7.1. triclopsGetBaseline ... *60*
6.7.2. triclopsGetCameraConfiguration ... *61*
6.7.3. triclopsGetDeviceConfiguration ... *61*
6.7.4. triclopsGetFocalLength ... *62*
6.7.5. triclopsGetImageCenter ... *62*

6.7.6. triclopsGetSerialNumber *63*
6.7.7. triclopsSetCameraConfiguration *63*
6.8. 3D 63
6.8.1. triclopsCreateImage3d *63*
6.8.2. triclopsDestroyImage3d *64*
6.8.3. triclopsExtractImage3d *64*
6.8.4. triclopsExtractWorldImage3d *65*
6.8.5. triclopsGetTransformFromFile *65*
6.8.6. triclopsGetTriclopsToWorldTransform *66*
6.8.7. triclopsRCD16ToWorldXYZ *66*
6.8.8. triclopsRCD16ToXYZ *67*
6.8.9. triclopsRCD8ToWorldXYZ *68*
6.8.10. triclopsRCD8ToXYZ *69*
6.8.11. triclopsRCDFloatToWorldXYZ *70*
6.8.12. triclopsRCDFloatToXYZ *71*
6.8.13. triclopsRCDMappedToWorldXYZ *71*
6.8.14. triclopsRCDMappedToXYZ *72*
6.8.15. triclopsRCDToWorldXYZ *73*
6.8.16. triclopsRCDToXYZ *74*
6.8.17. triclopsSetTriclopsToWorldTransform *75*
6.8.18. triclopsWorldXYZToRCD *76*
6.8.19. triclopsWriteTransformToFile *77*
6.8.20. triclopsXYZToRCD *77*
6.9. IMAGE BUFFER OPERATIONS 78
6.9.1. triclopsSetColorImageBuffer *78*
6.9.2. triclopsSetImage16Buffer *79*
6.9.3. triclopsSetImageBuffer *79*
6.9.4. triclopsSetPackedColorImageBuffer *80*
6.9.5. triclopsUnsetColorImageBuffer *80*
6.9.6. triclopsUnsetImage16Buffer *81*
6.9.7. triclopsUnsetImageBuffer *81*
6.9.8. triclopsUnsetPackedColorImageBuffer *82*
6.10. IMAGE I/O OPERATIONS 83
6.10.1. triclopsReadImage16Extra *83*
6.10.2. triclopsReadImageExtra *83*
6.10.3. triclopsSaveColorImage *84*
6.10.4. triclopsSaveImage16Extra *85*
6.10.5. triclopsSaveImageExtra *85*
6.10.6. triclopsSavePackedColorImage *86*
6.11. RECTIFICATION AND CAMERA GEOMETRY 87
6.11.1. triclopsGetResolution *87*
6.11.2. triclopsGetSourceResolution *87*
6.11.3. triclopsSetResolution *88*
6.11.4. triclopsSetResolutionAndPrepare *88*
6.11.5. triclopsSetSourceResolution *89*
6.12. RECTIFICATION 90

6.12.1. triclopsGetLowpass ... *90*
6.12.2. triclopsGetRectify ... *90*
6.12.3. triclopsGetRectImgQuality ... *90*
6.12.4. triclopsPreprocess ... *91*
6.12.5. triclopsRectify ... *91*
6.12.6. triclopsRectifyColorImage ... *92*
6.12.7. triclopsRectifyPackedColorImage ... *93*
6.12.8. triclopsRectifyPixel ... *93*
6.12.9. triclopsSetLowpass ... *94*
6.12.10. triclopsSetRectify ... *94*
6.12.11. triclopsSetRectImgQuality ... *94*
6.12.12. triclopsUnrectifyPixel ... *95*
6.13. TRICLOPS CONTEXT MANIPULATION ... 96
6.13.1. triclopsCopyContext ... *96*
6.13.2. triclopsDestroyContext ... *96*
6.13.3. triclopsGetDefaultContextFromFile ... *97*
6.13.4. triclopsWriteCurrentContextToFile ... *97*
6.13.5. triclopsWriteDefaultContextToFile ... *98*
6.14. UNGROUPED OBJECTS ... 98
6.14.1. TRICLOPS_VERSION ... *98*
7. CONTACTING POINT GREY RESEARCH ... 99

1. Introduction

This chapter presents the features of the Triclops Stereo Vision Software Development Kit (SDK). It also gives the reader pointers to other resources useful for mastering stereo vision technology.

1.1. Stereo Vision Technology

Stereo vision technology allows range measurement using triangulation between slightly offset cameras. The Bumblebee® Stereo Vision Family of Products consists of a two or three-camera module and a software system that performs range measurements. The camera module generates side-by-side images that are digitized and transferred to the host PC. The software system analyzes these "stereo" images and establishes correspondence between pixels in each image. Based on the camera's geometry and the correspondences between pixels in the images, it is possible to determine the distance to points in the scene. While "stereo" images appear quite similar, closer inspection reveals a shift between closer objects and those that are further away. Based on the amount of shift, the system is able to determine the distance to the objects in the scene and produce a depth image. The Triclops Stereo Vision SDK is the software component in the system that takes in side-by-side image and camera information to produce the above mentioned image shift information and ultimately scene depth.

This section was written to give the reader intuition about the functionality of the system. A more detailed description of the process and programming details can be obtained by consulting the programming examples provided with the Triclops Stereo Vision SDK and by contacting your support representative.

Who Should Read this Manual

This manual is intended for personnel responsible for installing the Triclops Stereo Vision SDK and for programmers developing applications with the Triclops system.

Installers must have a working knowledge of the Windows operating system at the system administrator level.

Developers will find familiarity with the concepts of stereo vision very useful in utilizing the Triclops system and making sense of the stereo results. For a general background in stereo vision, see 4. Stereo Vision Details.

System Requirements

To use the Triclops Stereo Vision SDK, you must have a Windows or Linux running PC with:

- 1 gigahertz (GHz) or faster 32-bit (x86) or 64-bit (x64) processor
- 1 gigabyte (GB) RAM (32-bit) or 2 GB RAM (64-bit)
- 1 Firewire host card (either S400 for Bumblebee ® 2 or S800 for Bumblebee ® XB3)

Online Resources

For the latest demos and sample code written for the Triclops Stereo Vision SDK, visit our web page at:
http://www.ptgrey.com

2. Installing the Software

This chapter discusses installing the software and setting up the operating system.

2.1. Installation of Triclops

The Triclops software system is currently distributed online or on a CD-ROM disk. Please consult the README file on the disk or in the downloaded package for a detailed description of the installation procedure.

To install the Triclops software on Windows:

1. Place the CD-ROM installation disk into your CD-ROM drive or locate the install package.
2. Run the installer that is appropriate for your Operating System type (32 or 64 bit). It is recommended that you install the Triclops software in the default location suggested by the install shield. If not, the example projects will have to be changed to include the correct SDK paths for include and library files.

Once the installer finishes the following directory structure will be present on your system:

Subdirectory	Contents
Bin / Bin64	These subdirectory contains compiled examples and utility programs.

	It also contains the triclops.dll, which is the dynamic link library for triclops.lib.
Src	This subdirectory contains Triclops example source code. The Triclops example source code is discussed in 6. Triclops Application Programming Interface (API) Function Reference.
Include	This subdirectory contains the include files required to compile using the Triclops library. The primary include file is the triclops.h. This subdirectory also contains the pnmutils.h file, which includes utilities for reading and writing PGM and PPM format images.
Lib	This subdirectory contains the Triclops and pnmutils libraries.
Doc	This subdirectory contains the Triclops Manual.

2.2. Image Viewing

At this time it is a good idea to also install an image-viewing program. The Triclops library supports the PGM image format.

3. Getting Started

This chapter goes through a number of exercises designed to ensure that the system is properly installed and to give the user an initial feel for the system's capabilities.

This chapter is for Windows users

3.1. Running the Demo Program

After installing the Triclops SDK and an appropriate stereo device (i.e.: a Bumblebee® 2 or Bumblebee® XB3 stereo vision system), it is a good idea to test the device and the stereo installation by running the interactive demonstration programs. Please refer to the documentation for your stereo vision system installation for a description of the different demonstration programs available and their operation.

For further information on the meaning of stereo parameters that can be set in the demonstration programs, please refer to 4. Stereo Vision Details.

3.2. Compiling a Triclops Program

Now that you know that the system is installed correctly, you can try compiling a sample program and verify that the Triclops library and include files are properly set. For this step, we assume that your development environment is Microsoft Visual Studio. In the following steps, it is assumed that you can locate the installed files on your system.

1. Copy contents of the SRC folder into a new folder for which you have full privileges.
2. Start Microsoft Visual Studio and open the "examples.sln" solution in your newly created directory.
3. Set the active project to the "stereo" project.
4. Build the program.
5. Run the program.
6. View the images, "disparity.pgm", "disparity16.pgm" and "rectified.pgm" that are created by the program in your newly created directory with an image viewer that supports the PGM file format.

This program reads a raw stereo image from a file, performs stereo processing and saves the gray-scale images and depth map into separate files.

4. Stereo Vision Details

This chapter presents an overview of stereo vision technology. After reading this chapter you will have a better understanding of the data flow in the library and all tunable parameters. This will allow you to customize the system to particular tasks.

The purpose of stereo vision is to perform range measurements based on images obtained from slightly offset cameras. There are three steps in performing stereo processing:

Establish correspondence between image features in different views of the scene.
Calculate the relative displacement between feature coordinates in each image.
Determine the 3D location of the feature relative to the cameras, using the knowledge of the camera geometry.

Consider the following example. Figure 1 shows an image pair obtained from the horizontally displaced cameras of the Triclops camera module. We can identify two points A and B in both images. The point A_{left} corresponds to the point A_{right}. Similarly, point B_{left} corresponds to the point B_{right}.

Figure 1: Example of matching points between stereo images

Using a ruler, if you measure out the horizontal distance between the left edge of the images and the points, you will find that distances in the left image are greater than the distance to the corresponding point in the right image. For example, the distance to the phone handle from the left edge of the image is greater than the distance to the phone

handle in the right image. Based on this distance (also called disparity) it is possible to determine the distance to the phone handle from the camera module.

We will define the disparity as the difference between the coordinates of the same features in the left and right image. You will find that the distances from the top of the image to the matching features are exactly the same in both images. This is because the cameras are horizontally aligned, therefore only the horizontal displacement is relevant.

Disparity for the feature A will be defined as $D(A) = x(A_{left}) - x(A_{right})$ and the disparity of point B will be derived as $D(B) = x(B_{left}) - x(B_{right})$, where $x(A_{left})$ is the x coordinate of the point A_{left}.

If you calculated D(A) and D(B) you will find that $D(B) > D(A)$ this indicated that point B in the scene is closer than point A.

4.1. Establishing Correspondence

The Triclops library establishes correspondence between images using the Sum of Absolute Differences correlation method. The intuition behind the approach is to do the following:

1. For every pixel in the image
2. Select a neighborhood of a given square size from the reference image
3. Compare this neighborhood to a number of neighborhoods in the other image (along the same row)
4. Select the best match
5. End

Comparison of neighborhoods or masks is done using the following formula:

$$\min_{d=d_{\min}}^{d_{\max}} \sum_{i=-\frac{m}{2}}^{\frac{m}{2}} \sum_{j=-\frac{m}{2}}^{\frac{m}{2}} \left| \mathrm{I}_{right}[x+i][y+j] - \mathrm{I}_{left}[x+i+d][y+j] \right|$$

where:

$d_{\min}$ and $d_{\max}$ are the minimum and maximum disparities.

m is the mask size.

I_{right} and I_{left} are the right and left images.

Equation 1: Sum of absolute differences

4.2. Calculating Distances

The distances from the cameras are determined using the displacement between images and the geometry of the cameras. The position of the matched feature is a function of the

displacement, the focal length of the lenses, resolution of the CCD and the displacement between cameras.

The Triclops library provides a function that converts depth maps into distance images.

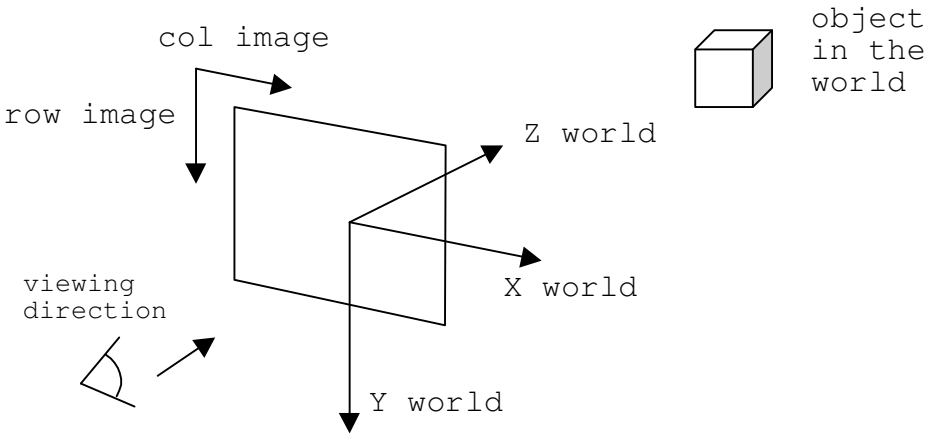

Figure 2: Image and world coordinate systems in the Triclops library

Figure 7 illustrates the coordinate system in which the Triclops library represents images and the world measurements. The origin of the image is in the top left corner of the upright image. The origin of the world measurements is in the pinhole of the reference camera.

4.3. Triclops Library Data Flow

Figure 3 shows the data flow in the Triclops library. The library takes raw images obtained from the Triclops camera module and produces depth images. There are two main processing blocks in the library. The first processing block is the image pre-processing block that applies a low-pass filter, rectifies the images and performs edge detection. The second processing block does stereo matching, validation of results and subpixel interpolation. The result of the library is a depth image.

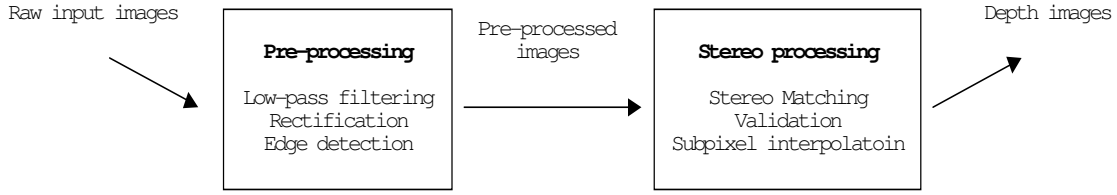

Figure 3: Triclops library data flow

4.4. Preprocessing

The pre-processing module of the Triclops library prepares the raw images for stereo processing. Pre-processing allows specification of the processing resolution, and the following functionality:
Low-Pass Filtering

In order to rectify the images it is important to smooth them. If image rectification is done it is a good idea to turn on the low pass filtering. Rectification can still be done with low pass filtering disabled but the rectified images will exhibit aliasing effects. The user has the option of disabling the low-pass filtering in order to speed up processing.

4.5. Rectification

Rectification is the process of correcting input images for the distortions of the lenses. Lenses often cause distortions in raw images. For example, straight lines in the scene will often appear curved in the raw images. This effect will be particularly evident in the corners of the images. Rectified images will be corrected for these kinds of distortions.

Further, rectified images will be corrected so that the rows of images digitized from horizontally displaced cameras are aligned, and similarly that the columns of images obtained from vertically displaced cameras are aligned. Without this feature, searching along the rows and columns will not produce the correct results.

4.6. Edge Detection

Edge detection is an optional feature that allows matching on the changes in the brightness rather than the absolute values of the pixels in the images. This feature is useful because the cameras in the Triclops camera module have auto gain control. If the auto gains in the cameras do not change identically, the absolute brightness between images may not be the same. While absolute brightness are not the same, the change in the intensity stays constant. Therefore, using edge detection will help in environments where the lighting conditions change significantly.

While edge detection may improve results, there is an additional processing cost that is associated with it. Therefore the user needs to evaluate the result improvement when choosing to turn edge detection on.

Note that validation is only available in the edge detection mode.

4.7. Stereo Processing

The stereo processing module applies the Sum of Absolute Differences algorithm described in previous sections. There are a number of parameters that determine the kind of depth image produced:

4.8. Disparity Range

Disparity range is the range of pixels that the stereo algorithm searches in order to find the best match. In the Triclops system a disparity of zero pixels corresponds to an infinitely far away object. The maximum disparity defines the closest position of an object that is to be determined. Users are allowed and encouraged to set the disparity range that is the most suitable for the task at hand. Reducing the disparity range will allow the system to run faster and will reduce the chance of a mismatching.

4.9. Correlation Mask

Correlation mask is a square neighborhood around the pixel that the system is trying to find the match for. The user is allowed to specify the size of the correlation mask. The correlation mask controls the coarseness of features compared between images. Larger masks will produce depth maps that are denser and smoother, however, they may lack precision in identifying the position of depth discontinuities. On the other hand smaller masks will produce sparser and more noisy depth images, but the localization of depth discontinuities will be much better.

To produce similar results the size of the mask must be proportional to the resolution of the images processed. Thus, in order to produce comparable results, a 5x5 mask on a 160x120-pixel image should be increased to a 9x9 mask for a 320x240-pixel image. Mask sizes must be odd numbers. Valid mask sizes are 3x3, 5x5, 7x7, and invalid mask sizes are 4x4, 6x6, 8x8. The Triclops is capable of supporting a maximum 15x15 mask to a 1x1 minimum. In addition, a new experimental function has been added: triclopsSetAnyStereoMask. This function allows the user to set any correlation mask size for stereo. This function is classified as an 'experimental' function, and care should be taken in its use. For more information, see the description of 'triclopsSetAnyStereoMask' in the Detailed Experimental Function Descriptions section.

4.10. Validation

In some cases, such as occlusions and lack of texture, it is not possible to establish correspondence between images. If the correspondence is not correct, the obtained measurements can not be correct. In order to avoid incorrect measurements, two validation methods are introduced:

Texture validation determines whether disparity values are valid based on levels of texture in the correlation mask. If the amount of texture is not sufficient to produce correct matches, the pixel will be declared invalid.
Uniqueness validation determines whether the best match for a particular pixel is significantly better than other matches within the correlation mask. Even if the correlation mask has enough texture, the correct match may not exist due to an occlusion. If the correlation result is not strong enough, the pixel will be declared invalid.

The user can specify two thresholds that control the strictness of validation - one for texture and one for uniqueness.

4.11. Better Calibration

The camera calibration has been improved to give greater overlap between cameras. This means a wider range of focal length lens can be used in the Triclops family of stereo vision cameras. Triclops SDK 2.2 is the first Triclops release that fully takes advantage of the new advances in calibration.

4.12. Subpixel Interpolation

The Triclops library allows matching between images to subpixel accuracy. The library takes advantage of the matching results of the neighboring pixels of the resulting disparity to determine an approximation that is within a fraction of a pixel. Accurate calibration between cameras allows an accuracy of 0.2 of a pixel.

This function marginally increases the computation time. If precise 3D position information is not required, it may be omitted.

4.13. Surface Validation

Triclops SDK 2.5 supports 'surface validation'. This is a filtering process designed to remove noise from the disparity image. The kind of noise removed with this process is 'spikes'. Spikes are characteristic of mismatches in correlation-based stereo vision. The spikes can often cover a connected region of many pixels. This noise is not zero-mean, random, evenly distributed or Gaussian. The result is that this noise is difficult to remove with standard filtering techniques, as it appears to be a valid signal, instead of noise. Surface validation is a method to validate regions of a disparity image based on an assumption that they must belong to a likely physical surface in the image. The method segments the disparity image into connected regions. Any region that is less than a given size, is suspect and removed from the disparity image. See also in 6. Triclops Application Programming Interface (API) Function Reference section: triclopsSetSurfaceValidation, triclopsGetSurfaceValidation, triclopsSetSurfaceValidationSize, triclopsGetSurfaceValidationSize,

triclopsSetSurfaceValidationDifference, triclopsGetSurfaceValidationDifference, triclopsSetSurfaceValidationMapping, triclopsGetSurfaceValidationMapping.

4.14. Subpixel Validation Mapping

In previous versions of the Triclops SDK, the validation mapping did not work when the subpixel interpolation flag was on. Now, validation mapping values are used during subpixel interpolation. Pixels which are invalid when performing subpixel validation are marked with values of 0xFF00 + mv, where 'mv' is the mapping value for the particular validation check that has failed.

4.15. Region of Interest and Subpixel

Unfortunately, although ROI processing does work with subpixel stereo, it does not improve the speed of processing. Therefore, we recommend that users only bother with ROI processing if they are not using the subpixel functionality of the stereo engine.

4.16. Disparity Mapping

Disparity mapping is a method to automatically scale the output disparity image between some fixed 'mapping' values. This is a convenience, especially for demos when one wants the disparity images to be scaled within the entire display range of brightness values. The problem with disparity mapping is that it can cause a lot of trouble when trying to use mapped information to extract 3D information. The RCDToXYZ family of functions are designed to extract 3D information from disparity values. If these functions have to first correct for disparity mapping, a significant performance reduction has incurred.

To make it easy to simply turn off and ignore disparity mapping, we have added a new flag for the TriclopsContext, the DisparityMappingOn flag. This is accessed through triclopsSet/GetDisparityMappingOn() functions. The default value will be 'false'. This means that most users of the Triclops SDK can ignore disparity mapping. Users who are using disparity mapping must change their programs to call triclopsSetDisparityMappingOn(context, 1) to enable disparity mapping.

Disparity mapping does not currently work for subpixel stereo, and Point Grey Research, Inc. (PGR) currently has no plans to extend disparity mapping to subpixel stereo. PGR would like to discourage users from depending on this functionality as this may be removed in later versions of the SDK.

5. Programming with the Triclops Application Programming Interface (API)

This chapter presents the Triclops API and explains the engine behind the functions. Several programming examples are presented in order to illustrate the API.

5.1. Programming with the Triclops Stereo Vision SDK

The goal of the Triclops Stereo Vision SDK is to provide the user with accurate and fast depth map generation. Therefore, the ultimate result of the Triclops Application Programming Interface (API) is a depth map. Depth maps can be produced in a number of different ways. Further characteristics of depth maps may vary depending on a number of settings. Triclops system allows specifying the following characteristics of the stereo processing:

Stereo Parameters	Description of the Parameter
Image size/resolution	The API allows the user to specify the size of the image that is to be used as the initial gray-scale image.
Disparity range	Allows the user to specify the range that distance measurements are to be done in.
Mask size	Allows the user to specify the coarseness of features that are to be matched between images.
Preprocessing	Specifies whether matching should be done on gray-scale or preprocessed images, such as edge images.
Validation	Specifies the methods used for verifying the correctness of matched-between images.
Regions Of Interest	Specifies the region of the image that processing should be done on. This feature allows processing speedup.
Subpixel Interpolation	This feature allows establishing correspondences to subpixel accuracy allowing generation of more precise distance measurements.

For detailed descriptions of stereo vision parameters please refer to 4. Stereo Vision Details.

5.2. Triclops Context

The Triclops software system allows specifying all characteristics of stereo processing discussed above. Furthermore, the software system allows the specification of multiple stereo processing that may occur on a single set of images. To enable efficient stereo processing of different kinds on the same set of images, the concept of Triclops Contexts is introduced.

By using the Triclops context it is possible to encapsulate all of the information required for a specific kind of stereo processing. Furthermore, multiple Triclops contexts allow for the sharing of data and processing with minimal effort on the user's part. Triclops contexts store camera configuration, parameters of stereo processing, input data and results.

A Triclops context must first be initialized using the configuration of the camera module. Configuration contains information about the number and geometry of the cameras, as well as the intrinsic and extrinsic parameters of the cameras.

A Triclops context is then configured for the specific kind of stereo processing. Parameters such as the processing resolution, disparity range, validation, and subpixel interpolation may be specified.

Next, a Triclops context must be assigned images that are to be processed by the stereo kernel. This is done by passing into a context information obtained from the stereo device, or by loading the image information from the file.

The Triclops context is then used for performing image pre-processing and the stereo processing. The results of stereo processing can then be retrieved from the context.

5.3. Triclops Example Source Code

In order to quickly understand the Triclops library, a number of examples are provided with the software distribution. Please consult these to get a primary idea of how the SDK functions. For further assistance please contact your product service representative.

6. Triclops Application Programming Interface (API) Function Reference

This chapter presents a detailed description of each function in the API.

6.1. Enumerations

6.1.1. TriclopsCamera

Declaration

```
enum TriclopsCamera
{
  TriCam_REFERENCE,
  TriCam_RIGHT,
  TriCam_TOP,
  TriCam_LEFT,

  TriCam_L_RIGHT      = TriCam_RIGHT,
  TriCam_L_TOP        = TriCam_TOP,
  TriCam_L_LEFT       = TriCam_LEFT,

  TriCam_COLOR,
  TriCam_L_COLOR = TriCam_COLOR

}
```

This enumerated type identifies individual cameras given a specific camera configuration.

Elements

TriCam_COLOR	
TriCam_L_COLOR	
TriCam_L_LEFT	
TriCam_L_RIGHT	These are kept here as legacy code for now... should be replaced in user code to not include the configuration specific "_L_" These values may be phased out in future releases of Triclops SDK
TriCam_L_TOP	
TriCam_LEFT	

TriCam_REFERENCE	
TriCam_RIGHT	
TriCam_TOP	

Remarks
TriCam_COLOR is only for use with the Color Triclops (may be phased out in future releases of Triclops SDK).

6.1.2. TriclopsCameraConfiguration

Declaration
```
enum TriclopsCameraConfiguration
{
  TriCfg_L = 0,
  TriCfg_2CAM_HORIZONTAL = 1,
  TriCfg_2CAM_HORIZONTAL_NARROW = 1,
  TriCfg_2CAM_VERTICAL = 2,
  TriCfg_2CAM_HORIZONTAL_WIDE = 3,
  TriCfg_MAX_VALUE = 3

}
```
This enumerated type defines the camera configuration. The symbols in the table represent the cameras as they would be seen when the camera module is viewed from the front. This type is either read from the camera or is set manually to indicate 2-Camera mode.

Elements

TriCfg_2CAM_HORIZONTAL	2 Camera Unit or 3 Camera Unit in 2 camera stereo mode.
TriCfg_2CAM_HORIZONTAL_NARROW	12cm baseline for BB2 or XB3 2 Camera Vertical Unit or 3 Camera Unit in 2 camera vertical mode.
TriCfg_2CAM_HORIZONTAL_WIDE	24cm baseline for XB3
TriCfg_2CAM_VERTICAL	
TriCfg_L	** Obsolete **
TriCfg_MAX_VALUE	

6.1.3. TriclopsError

Declaration
```
enum TriclopsError
{
  TriclopsErrorOk = 0,
  TriclopsErrorNotImplemented,
  TriclopsErrorInvalidSetting,
  TriclopsErrorInvalidContext,
```

TriclopsErrorInvalidCamera,
TriclopsErrorInvalidROI,
TriclopsErrorInvalidRequest,
TriclopsErrorBadOptions,
TriclopsErrorCorruptConfigFile,
TriclopsErrorNoConfigFile,
TriclopsErrorUnknown,
TriclopsErrorNonMMXCpu,
TriclopsErrorInvalidParameter,
TriclopsErrorSurfaceValidationOverflow,
TriclopsErrorCorruptTransformFile,
TriclopsErrorSystemError,
TriclopsErrorFileRead,
TriclopsErrorCorruptPGRComment,
TriclopsErrorDeprecated

}

All Triclops functions return an error value that indicates whether an error occurred, and if so what error. The following enumerated type lists the kinds of errors that may be returned.

Elements

TriclopsErrorBadOptions	Some options are illegal or contradictory.
TriclopsErrorCorruptConfigFile	The configuration file is corrupted, missing mandatory fields, or is for the wrong Triclops version.
TriclopsErrorCorruptPGRComment	The PGR comment in a PGM header was corrupted
TriclopsErrorCorruptTransformFile	An error has occurred during a system call. Check 'errno' to determine the reason. This includes the system running out of memory.
TriclopsErrorDeprecated	This Triclops functionality has been deprecated
TriclopsErrorFileRead	Error reading in a file.
TriclopsErrorInvalidCamera	The specified camera is not valid for this camera configuration.
TriclopsErrorInvalidContext	The TriclopsContext passed in was corrupt or invalid.
TriclopsErrorInvalidParameter	An invalid parameter has been passed.
TriclopsErrorInvalidRequest	Given the current specified options, the request is not valid. For example, requesting the disparity image from a context that has

	not executed 'triclopsStereo()'.
TriclopsErrorInvalidROI	An impossible Region Of Interest was specified. For example, a region of interest with negative height or width.
TriclopsErrorInvalidSetting	User specified input to the function that was not a valid value for this function.
TriclopsErrorNoConfigFile	Can not find the configuration file.
TriclopsErrorNonMMXCpu	A function that requires MMX was called on a non-MMX enabled CPU.
TriclopsErrorNotImplemented	User requested an option that is not yet implemented.
TriclopsErrorOk	Function succeeded.
TriclopsErrorSurfaceValidationOverflow	A call to TriclopsSetSurfaceValidation has caused an overflow.
TriclopsErrorSystemError	Can not find the transform file or its contents is invalid
TriclopsErrorUnknown	An indeterminable error occurred.

6.1.4. TriclopsImage16Type

Declaration
```
enum TriclopsImage16Type
{
  TriImg16_DISPARITY = 0

}
```
This enumerated type defines the various 16bit image types.

Elements

TriImg16_DISPARITY	Disparity Image: This is the 16-bit resultant depth image after stereo processing with subpixel on.

6.1.5. TriclopsImageType

Declaration
```
enum TriclopsImageType
{
  TriImg_DISPARITY = 0,
  TriImg_RAW,
  TriImg_RECTIFIED,
  TriImg_EDGE,
```

}
The enumerated type TriclopsImageType indicates what kind of image is either being requested or returned. It also indicates the size of each pixel in the image.

Elements

TriImg_DISPARITY	Disparity image: This is the resultant depth image after stereo processing.
TriImg_EDGE	Edge image: A Bandpass filter has been applied to this image. This image has values that range about 128. A value of 128 indicates no edge. Values greater and less than 128 indicate an edge with strength relative to the difference between 128 and the pixel value.
TriImg_RAW	Raw unrectified image: This is an image with the aspect ratio that was supplied by the input. There is no correction for lens distortion or camera misalignment.
TriImg_RECTIFIED	Rectified image: This is an image that has been corrected for lens distortion and camera misalignment to fit a pinhole camera model.

6.1.6. TriclopsInputType

Declaration
```
enum TriclopsInputType
{
  TriInp_NONE,
  TriInp_RGB_32BIT_PACKED,
  TriInp_RGB

}
```
This enumerated type identifies the format of the input to the Triclops library. This input has generally either been read from a set of files (for off line processing) or has just been delivered by a frame grabber. There are two formats of input data that are currently being supported. RGB format indicates that the data is supplied with a separate buffer for each R, G and B channel. RGB packed indicates that the 3 channels are interleaved.

Elements

TriInp_NONE	This is used to mark that the input has not yet been set.
TriInp_RGB	An array of separated bands with the following ordering: [R R R....][G G G...][B B B...].
TriInp_RGB_32BIT_PACKED	An array of pixels with color bands interleaved in the following order: [B G R U] [B G R U].

6.2. Types

6.2.1. DisparityImageInfo

Declaration
```
typedef struct
{
  TriclopsRectImgQuality   rectQuality;
  bool          edgeCorrelationOn;
  int           edgeMaskSize;
  bool          lowpassOn;
  TriclopsStereoQuality    stereoQuality;
  TriclopsCameraConfiguration    camConfig;
  int           stereoMaskSize;
  int           minDisparity;
  int           maxDisparity;
  int           nDisparityOffset;
  bool          disparityMappingOn;
  int           minDispMapping;
  int           maxDispMapping;
  bool          textureValidOn;
  double        textureValidThreshold;
  unsigned char     textureValidMapping;
  bool          uniqueValidOn;
  double        uniqueValidThreshold;
  unsigned char     uniqueValidMapping;
  bool          backForthValidOn;
  unsigned char     backForthValidMapping;
  bool          surfaceValidOn;
  double        surfaceValidDiff;
  int           surfaceValidSize;
  unsigned char     surfaceValidMapping;
} DisparityImageInfo
```
Information that is stored in a Pgm file created using triclopsSaveImageExtra when passed the TriclopsImageType TriImg_DISPARITY as the second parameter.

6.2.2. EdgeImageInfo

Declaration
```
typedef struct
{
  TriclopsRectImgQuality rectQuality;
  int edgeMaskSize;
  bool  lowpassOn;
} EdgeImageInfo
```

Information that is stored in a Pgm file created using triclopsSaveImageExtra when passed the TriclopsImageType TriImg_EDGE as the second parameter.

6.2.3. RectifiedImageInfo

Declaration
typedef struct
{
  TriclopsRectImgQuality rectQuality;
} RectifiedImageInfo
Information that is stored in a Pgm file created using triclopsSaveImageExtra when passed the TriclopsImageType TriImg_RECTIFIED as the second parameter.

6.2.4. TriclopsBool

Declaration
typedef int TriclopsBool
Definition for Boolean variables.

6.2.5. TriclopsColorImage

Declaration
struct TriclopsColorImage
{
  int nrows;
  int ncols;
  int rowinc;
  unsigned char* red;
  unsigned char* green;
  unsigned char* blue;

}
This structure is used for image output from the Triclops system for color images. The structure is the same as TriclopsImage except that the data field is replaced by three color bands; 'red', 'green' and 'blue'. Each band is a complete image for that particular color band. In this case, rowinc is the row increment for each color band.

Elements

blue	The pixel data for blue band of the image.
green	The pixel data for green band of the image.
ncols	The number of columns in the image.
nrows	The number of rows in the image.
red	The pixel data for red band of the image.
rowinc	The row increment of the image.

6.2.6. TriclopsContext

Declaration
typedef void* TriclopsContext
Triclops context is a pointer to an internal structure that maintains all image and bookkeeping information necessary to the Triclops library. It is the container for all parameters

6.2.7. TriclopsImage

Declaration
```
struct TriclopsImage
{
  int   nrows;
  int   ncols;
  int   rowinc;
  unsigned char* data;

}
```
This structure is used both for image input and output from the Triclops system.

Elements

data	The area for the pixel data. This must be numRows * numCols bytes large.
ncols	The number of columns in the image.
nrows	The number of rows in the image.
rowinc	This is the pitch, or row increment for the image. Data must be contiguous within each row, and the rows must be equally spaced. Rowinc indicates the number of bytes between the beginning of a row and the beginning of the following row.

6.2.8. TriclopsImage16

Declaration
```
struct TriclopsImage16
{
  int          nrows;
  int          ncols;
  int          rowinc;
  unsigned short*   data;

}
```
This structure is used for image output from the Triclops system for image types that require 16-bits per pixel. This is the format for subpixel interpolated images. The structure is identical to the TriclopsImage structure except that the data contains unsigned shorts rather than unsigned chars. Rowinc is still the number of bytes between the beginning of a row and the beginning of the following row (NOT number of pixels).

Elements

data	The pixel data of the image.
ncols	The number of columns in the image.
nrows	The number of rows in the image.
rowinc	The number row increment of the image.

6.2.9. TriclopsImage3d

Declaration

```
struct TriclopsImage3d
{
  int nrows;
  int ncols;
  int rowinc;
  TriclopsPoint3d*     points;

}
```

This structure defines the format of an image consisting of 3d points.

Elements

ncols	The number of columns in the image.
nrows	The number of rows in the image.
points	The area for the pixel data. This must be numRows * numCols bytes large.
rowinc	The number of bytes between the beginning of a row and the beginning of the following row

6.2.10. TriclopsImageInfo

Declaration

```
typedef struct
{
  bool commentFound;
  char product[100];
  int serialNumber;
  TriclopsCamera nCamera;
  union
  {
    TriclopsImageType    imageType;
    TriclopsImage16Type  image16Type;
  } type;
  union
  {
    RectifiedImageInfo rectified;
    EdgeImageInfo   edge;
    DisparityImageInfo disparity;
  } info;
```

} TriclopsImageInfo
This structure is used to hold any image information available when an image is read in to the triclopsLibrary. It currently works with the function calls triclopsReadImage[16]Extra. Different information is available depending on the TriclopsImageType of the image -- that is the function of the union. The information will be available if the image was created using the triclopsSaveImage[16]Extra function.

6.2.11. TriclopsInput

Declaration
```
struct TriclopsInput
{
  TriclopsInputType inputType;
  TriclopsTimestamp timeStamp;
  int   nrows;
  int   ncols;
  int   rowinc;

  union
  {
    TriclopsInputRGB        rgb;
    TriclopsInputRGB32BitPacked   rgb32BitPacked;
  } u;

}
```
TriclopsInput structure contains image input to the Triclops library. The library accepts two formats: 32-bit packed, and RGB separate buffer inputs. Field u contains a pointer to one of the two typed structures that contain the image data. The inputType field indicates which type of input is actually present in the structure.

Elements

inputType	The input type indicating packed or unpacked data.
ncols	The number of columns in the input images.
nrows	The number of rows in the input images.
rowinc	The row increment of the input images.
timeStamp	The timestamp on the image data (generated by the camera.)
u	The actual image data is either packed or unpacked and can be accessed with either TriclopsinputRGB or TriclopsInputRGB32BitPacked.

6.2.12. TriclopsInputRGB

Declaration
```
struct TriclopsInputRGB
{
  void*    red;
```

```
  void*    green;
  void*    blue;

}
```
This structure consists of three separate buffers, each containing nrows * ncols pixels for each of the RGB bands.

Elements

blue	The blue band.
green	The green band.
red	The red band.

6.2.13. TriclopsInputRGB32BitPacked

Declaration
```
struct TriclopsInputRGB32BitPacked
{
  void*    data;

}
```
This structure contains RGB packed data.

Elements

data	a pointer to an array of nrows*ncols*4 pixels. The pixels are organized in the following fashion [RGBU][RGBU] ...

6.2.14. TriclopsPackedColorImage

Declaration
```
struct TriclopsPackedColorImage
{
  int   nrows;
  int   ncols;
  int   rowinc;
  TriclopsPackedColorPixel*   data;

}
```
This structure defines the format of a 32bit packed color image.

Elements

data	A pointer to the pixel data.
ncols	The number of columns in the image.
nrows	The number of rows in the image.
rowinc	The number of bytes in each row of the image.

6.2.15. TriclopsPackedColorPixel

Declaration
```
struct TriclopsPackedColorPixel
{
  unsigned char     value[4];

}
```
This structure defines the format for a 32bit color pixel.

Elements

value	The 32 bit pixel data.

6.2.16. TriclopsPoint3d

Declaration
```
struct TriclopsPoint3d
{
  float point[3];

}
```
This structure defines a single 3d pixel. It is only used in the TriclopsImage3d structure.

Elements

point	The 3 values for the (x,y,z) point in order x = 0, y = 1, z = 2.

6.2.17. TriclopsRectImgQuality

Declaration
```
enum TriclopsRectImgQuality
{
  TriRectQlty_FAST,
  TriRectQlty_STANDARD,
  TriRectQlty_ENHANCED_1,
  TriRectQlty_ENHANCED_2
}
```
This enumerated type identifies

Elements

TriRectQlty_ENHANCED_1	
TriRectQlty_ENHANCED_2	
TriRectQlty_FAST	
TriRectQlty_STANDARD	

6.2.18. TriclopsROI

Declaration

```
struct TriclopsROI
{
  int   row;
  int   col;
  int   nrows;
  int   ncols;

}
```

This structure describes a Region Of Interest for selective processing of the input images.

Elements

col	The column value for the upper-left corner of the region of interest.
ncols	The width of the region of interest.
nrows	The height of the region of interest.
row	The row value for the upper-left corner of the region of interest.

6.2.19. TriclopsStereoQuality

Declaration

```
enum TriclopsStereoQuality
{
  TriStereoQlty_STANDARD,
  TriStereoQlty_ENHANCED
}
```

This enumerated type identifies stereo algorithm used to compute the disparity images. In general, the higher quality of algorithm chosen, the more processing time required to compute the result.

Elements

TriStereoQlty_ENHANCED	
TriStereoQlty_STANDARD	

6.2.20. TriclopsTimestamp

Declaration

```
struct TriclopsTimestamp
{
  long sec;
  long u_sec;

}
```

This structure describes the format by which time is represented in the Triclops Stereo Vision SDK.

Elements

sec	The number of seconds since the epoch.
u_sec	The number of microseconds within the second.

6.2.21. TriclopsTransform

Declaration
typedef struct
{
  double matrix[4][4];

} TriclopsTransform
A transformation matrix.

6.3. Debugging and Error Reporting

6.3.1. triclopsErrorToString

Declaration
char*
triclopsErrorToString(TriclopsError error)
Converts a Triclops error into a meaningful string.

This function returns a string that describes the given TriclopsError. This allows error reporting software to report meaningful error messages to the user.

Parameters

error	The value of a TriclopsError variable. Returns: char* - A string describing the nature of the TriclopsError.

See Also
TriclopsError

6.4. Validation Support

6.4.1. triclopsGetBackForthValidation

Declaration
TriclopsError
triclopsGetBackForthValidation(const TriclopsContext context,
                    TriclopsBool *on)
Gets the current back-forth validation setting.

This function is used to obtain the current setting of back-forth validation.

Parameters

context	TriclopsContext for the operation.
on	A pointer for the returned value of the current setting of the back-forth validation flag. Returns:

TriclopsErrorOk	Operation successful.

	TriclopsErrorInvalidContext	Context is not a valid TriclopsContext.	

See Also
triclopsSetBackForthValidation ()

6.4.2. triclopsGetBackForthValidationMapping

Declaration
TriclopsError
triclopsGetBackForthValidationMapping(const TriclopsContext context,
unsigned char *value)
Retrieves the back-forth validation threshold.

Parameters

context	The context.
value	A pointer that will return the current value of the back-forth validation mapping parameter. Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid.

See Also
triclopsSetBackForthValidationMapping()

6.4.3. triclopsGetStrictSubpixelValidation

Declaration
TriclopsError
triclopsGetStrictSubpixelValidation(TriclopsContext context,
TriclopsBool* on)
Retrieves the value that appears in the disparity image for pixels that fail uniqueness validation.

Parameters

context	The context.
on	A pointer to a Boolean that will contain the current setting. Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid.

See Also
triclopsSetStrictSubpixelValidation(), triclopsSetSubpixelInterpolation(), triclopsGetSubpixelInterpolation()

6.4.4. triclopsGetSubpixelValidationMapping

Declaration

TriclopsError
triclopsGetSubpixelValidationMapping(const TriclopsContext context,
unsigned char* value)
Retrieves the value that appears in the disparity image for pixels that fail subpixel validation.

Parameters

context	TriclopsContext for the operation.
value	A pointer that will hold the subpixel validation mapping value. Returns:

TriclopsErrorOk	Operation successful.
TriclopsErrorInvalidContext	Context is not a valid TriclopsContext.

See Also
triclopsSetSubpixelValidationMapping

6.4.5. triclopsGetSurfaceValidation

Declaration
TriclopsError
triclopsGetSurfaceValidation(const TriclopsContext context,
TriclopsBool* on)
Gets the current surface validation setting.

This function is used to obtain the current setting of surface validation.

Parameters

context	TriclopsContext for the operation.
on	A pointer for the returned value of the current setting of the surface validation flag. Returns:

TriclopsErrorOk	Operation successful.
TriclopsErrorInvalidContext	Context is not a valid TriclopsContext.

See Also
triclopsSetSurfaceValidation ()

6.4.6. triclopsGetSurfaceValidationDifference

Declaration
TriclopsError
triclopsGetSurfaceValidationDifference(
const TriclopsContext context,
float* diff)
Returns the maximum disparity difference between two adjacent pixels that will still allow the two pixels to be considered part of the same surface.

Parameters

context	TriclopsContext for the operation.
diff	A pointer that returns the current value of the surface validation difference

	parameter. Returns: TriclopsErrorOk — Operation successful. TriclopsErrorInvalidContext — Context is not a valid TriclopsContext.

6.4.7. triclopsGetSurfaceValidationMapping

Declaration
TriclopsError
triclopsGetSurfaceValidationMapping(const TriclopsContext context,
unsigned char* value)
Retrieves the current surface validation mapping.

This function returns the current setting for the surface validation parameter.

Parameters

context	TriclopsContext for the operation.
value	A pointer that will return the current value of the surface validation mapping parameter. Returns: TriclopsErrorOk — Operation successful. TriclopsErrorInvalidContext — Context is not a valid TriclopsContext.

6.4.8. triclopsGetSurfaceValidationSize

Declaration
TriclopsError
triclopsGetSurfaceValidationSize(const TriclopsContext context,
int* size)
Retrieves the current validation size.

This function is used to extract the surface size parameter for surface validation.

Parameters

context	TriclopsContext for the operation.
size	A pointer that returns the current value of the size parameter. Returns: TriclopsErrorOk — Operation Successful. TriclopsErrorInvalidContext — Context is not a valid TriclopsContext.

See Also
triclopsSetSurfaceValidationSize

6.4.9. triclopsGetTextureValidation

Declaration
TriclopsError
triclopsGetTextureValidation(const TriclopsContext context,
TriclopsBool* on)

Retrieves the state of the texture validation flag.

Parameters

context	The context.
on	Container that receives the texture validation flag value. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
TriclopsSetTextureValidation()

6.4.10. triclopsGetTextureValidationMapping

Declaration
TriclopsError
triclopsGetTextureValidationMapping(const TriclopsContext context,
unsigned char* value)
Gets the value that appears in the disparity image for pixels that fail texture validation.

Parameters

context	The TriclopsContext to get the texture validation mapping from.
value	The current texture validation mapping setting. `Returns:`

TriclopsErrorOk	Upon the successful completion of the operation.

6.4.11. triclopsGetTextureValidationThreshold

Declaration
TriclopsError
triclopsGetTextureValidationThreshold(const TriclopsContext context,
float* value)
Retrieves the texture validation threshold.

Parameters

context	The context.
value	The location for the retrieved threshold. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
TriclopsSetTextureValidationThreshold()

6.4.12. triclopsGetUniquenessValidation

Declaration

TriclopsError
triclopsGetUniquenessValidation(const TriclopsContext context,
TriclopsBool* on)
Retrieves the state of the uniqueness validation

Parameters

context	The context.
on	A pointer to a Boolean. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
TriclopsSetUniquenessValidation()

6.4.13. triclopsGetUniquenessValidationMapping

Declaration
TriclopsError
triclopsGetUniquenessValidationMapping(
const TriclopsContext context,
unsigned char* value)
Retrieves the value that appears in the disparity image for pixels that fail uniqueness validation.

Parameters

context	The context.
value	A pointer to an unsigned char that will contain the current value of the mapping. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
TriclopsSetUniquenessValidationMapping()

6.4.14. triclopsGetUniquenessValidationThreshold

Declaration
TriclopsError
triclopsGetUniquenessValidationThreshold(
const TriclopsContext context,
float* value)
Retrieves the uniqueness validation threshold.

Parameters

context	The context.
value	A pointer to a float indicating the current threshold. Returns:

TriclopsErrorOk	The operation succeeded.

	InvalidContext	The input context was invalid.

See Also
TriclopsSetUniquenessThreshold()

6.4.15. triclopsSetBackForthValidation

Declaration
TriclopsError
triclopsSetBackForthValidation(TriclopsContext context,
TriclopsBool on)
Enables or disables back-forth validation.

This function is used to enable or disable back-forth validation.

Parameters

context	TriclopsContext for the operation.
on	A Boolean flag indicating if back-forth validation should be enabled or disabled. Returns:

TriclopsErrorOk	Operation successful.
TriclopsErrorInvalidContext	Context is not a valid TriclopsContext.

See Also
triclopsGetSurfaceValidation ()

6.4.16. triclopsSetBackForthValidationMapping

Declaration
TriclopsError
triclopsSetBackForthValidationMapping(TriclopsContext context,
unsigned char value)
Sets the value that appears in the disparity image for pixels that fail back-forth validation.

Sets the back-forth validation mapping value. This is the value that is assigned to pixels in the disparity image that fail the back-forth validation test. The range is between 0 and 255.

Parameters

context	The context.
value	The value to which failing pixels are mapped. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsGetBackForthValidationMapping()

6.4.17. triclopsSetStrictSubpixelValidation

Declaration
TriclopsError
triclopsSetStrictSubpixelValidation(TriclopsContext context,
TriclopsBool on)
Sets the state of the subpixel validation setting.

The strict subpixel validation option is used when the subpixel interpolation flag is turned on. Strict subpixel validation enables a more restrictive validation method for subpixel validation. With strict subpixel validation on, there will be less data in the output image, but it will be more reliable.

Parameters

context	The context
on	A Boolean value indicating whether validation is on or off. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsGetStrictSubpixelValidation(), triclopsSetSubpixelInterpolation(), triclopsGetSubpixelInterpolation()

6.4.18. triclopsSetSubpixelValidationMapping

Declaration
TriclopsError
triclopsSetSubpixelValidationMapping(TriclopsContext context,
unsigned char value)
Sets the value that appears in the disparity image for pixels that fail subpixel validation.

Strict subpixel validation verifies that the disparity values contribute to the subpixel interpolation make sense. By setting the mapping value to 0x??, an invalid pixel that failed this validation step will be marked 0xFF??.

Parameters

context	TriclopsContext for the operation.
value	The new subpixel validation mapping value. The range is between 0 and 255. Returns:

TriclopsErrorOk	The operation succeeded.
TriclopsErrorInvalidContext	Context is not a valid TriclopsContext.

6.4.19. triclopsSetSurfaceValidation

Declaration
TriclopsError
triclopsSetSurfaceValidation(TriclopsContext context,
TriclopsBool on)

Enables or disables surface validation.

This function is used to enable or disable surface validation.

Parameters

context	TriclopsContext for the operation.
on	A Boolean flag indicating if surface validation should be enabled or disabled. Returns:

TriclopsErrorOk	Operation successful.
TriclopsErrorInvalidContext	Context is not a valid TriclopsContext.

6.4.20. triclopsSetSurfaceValidationDifference

Declaration
TriclopsError
triclopsSetSurfaceValidationDifference(
TriclopsContext context,
float diff)
Set the maximum disparity difference between two adjacent pixels that will still allow the two pixels to be considered part of the same surface.

Parameters

context	TriclopsContext for the operation.
diff	The maximum disparity difference between two adjacent pixels that will still allow the two pixels to be considered part of the same surface. Returns:

TriclopsErrorOk	Operation successful.
TriclopsErrorInvalidContext	Context is not a valid TriclopsContext.
TriclopsErrorInvalidSetting	A negative difference was given.

6.4.21. triclopsSetSurfaceValidationMapping

Declaration
TriclopsError
triclopsSetSurfaceValidationMapping(TriclopsContext context,
unsigned char value)
Sets the current surface validation mapping.

Surface validation is a noise rejection method. By setting the mapping value to 0x??, an invalid pixel that failed this validation step will be marked 0xFF??.

Parameters

context	TriclopsContext for the operation.
value	Mapping value for pixels that fail surface validation. The range is between 0 and 255. Returns:

TriclopsErrorOk	Operation successful.
TriclopsErrorInvalidContext	Context is not a valid TriclopsContext

6.4.22. triclopsSetSurfaceValidationSize

Declaration
TriclopsError
triclopsSetSurfaceValidationSize(TriclopsContext context,
int size)
Sets the minimum number of pixels a surface can cover and still be considered valid.

This function is used to set the minimum number of pixels a surface can cover and still be considered valid. The larger the number is, the fewer surfaces will be accepted. The lower the number is, the more surfaces will be accepted. Common parameter values range from 100 to 500, depending on image resolution.

Parameters

context	TriclopsContext for the operation.
size	The minimum number of pixels a surface can cover and still be considered valid. Returns:

TriclopsErrorOk	Operation successful.
TriclopsErrorInvalidContext	Context is not a valid TriclopsContext.
TriclopsErrorInvalidSetting	A negative size was given.

6.4.23. triclopsSetTextureValidation

Declaration
TriclopsError
triclopsSetTextureValidation(const TriclopsContext context,
TriclopsBool on)
Turns texture validation on or off.

Sets the context texture validation flag. When set to true, texture-based validation is enabled. Pixels that do not pass the validation test will be marked with the texture validation mapping value in the disparity image.

Parameters

context	The context.
on	Texture validation flag. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsGetTextureValidation()

6.4.24. triclopsSetTextureValidationMapping

Declaration

TriclopsError
triclopsSetTextureValidationMapping(TriclopsContext context,
unsigned char value)
Sets the value that appears in the disparity image for pixels that fail texture validation.

Parameters

context	The TriclopsContext to set the texture validation mapping for.
value	The new value to map invalid pixels to. The range is from 0 to 255. Returns: TriclopsErrorOk — Upon the successful completion of the operation.

6.4.25. triclopsSetTextureValidationThreshold

Declaration
TriclopsError
triclopsSetTextureValidationThreshold(TriclopsContext context,
float value)
Sets the texture validation threshold.

Sets the texture validation threshold. This threshold allows one to tune the texture-based rejection of pixels. Values range from 0.0 (no rejection) to 128.0 (complete rejection) but good operating range is between 0.0 and 2.0.

Parameters

context	The context
value	The new texture validation threshold. Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid.

See Also
TriclopsGetTextureValidationThreshold()

6.4.26. triclopsSetUniquenessValidation

Declaration
TriclopsError
triclopsSetUniquenessValidation(TriclopsContext context,
TriclopsBool on)
Turns uniqueness validation on or off.

Turns uniqueness validation on or off. Uniqueness validation verifies that the match of a given pixel to its corresponding pixels in the top and left images is unique enough to be considered the definite correct match. If pixels at other disparities have almost as good matching to the given pixel as the best disparity, the pixel can be rejected as an unconfident match.

Parameters

context	The context.
on	A Boolean value indicating whether validation is on or off. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
TriclopsGetUniquenessValidation()

6.4.27. triclopsSetUniquenessValidationMapping

Declaration
TriclopsError
triclopsSetUniquenessValidationMapping(
TriclopsContext context,
unsigned char value)
Sets the value that appears in the disparity image for pixels that fail uniqueness validation.

Sets the unique matching validation mapping value. This is the value that is assigned to pixels in the disparity image that fail the uniqueness validation test. The range is between 0 and 255.

Parameters

context	The context.
value	The value to which failing pixels are mapped. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
TriclopsGetUniquenessValidationMapping()

6.4.28. triclopsSetUniquenessValidationThreshold

Declaration
TriclopsError
triclopsSetUniquenessValidationThreshold(
TriclopsContext context,
float value)
Sets the uniqueness validation threshold.

This function sets the uniqueness matching criteria threshold. This value can range from 0.0 to 10.0. The larger the number, the less rejection occurs. Common operating ranges are between 0.5 and 3.0.

Parameters

context	The context.
value	A float indicating the new threshold. Returns:

	TriclopsErrorOk	The operation succeeded.
	InvalidContext	The input context was invalid.

See Also
TriclopsGetUniquenessValidationThreshold()

6.5. General

6.5.1. triclopsBuildPackedTriclopsInput

Declaration
TriclopsError
triclopsBuildPackedTriclopsInput (int iCols,
int iRows,
int iRowInc,
unsigned long ulSeconds,
unsigned long ulMicroSeconds,
unsigned char* pDataPacked,
TriclopsInput* triclopsInput)
Uses the parameters passed in to build up a Triclops Input

This function takes a FlyCapture image and converts it to a triclops input. Depending on the camera (BB, BB2, XB3), it will deinterleave the images and create a Triclops Input of RGB type.

Parameters

iCols	The number of columns in the image
iRows	The number of rows in the image
iRowInc	The number of bytes in each row of the image
ulSeconds	The second value of the image timestamp
ulMicroSeconds	The microsecond value of the image timestamp
pDataPacked	An array to hold the packed data
triclopsInput	A TriclopsInput created by the FlyCapture image passed in. Returns: TriclopsErrorOk: The operation succeeded. TriclopsErrorFailed: The operation failed.

6.5.2. triclopsBuildRGBTriclopsInput

Declaration
TriclopsError
triclopsBuildRGBTriclopsInput (int iCols,
int iRows,
int iRowInc,
unsigned long ulSeconds,
unsigned long ulMicroSeconds,

unsigned char* pDataR,
unsigned char* pDataG,
unsigned char* pDataB,
TriclopsInput* triclopsInput)

Uses the parameters passed in to build up a Triclops Input

This function takes a FlyCapture image and converts it to a triclops input. Depending on the camera (BB, BB2, XB3), it will deinterleave the images and create a Triclops Input of RGB type.

Parameters

iCols	The number of columns in the image
iRows	The number of rows in the image
iRowInc	The number of bytes in each row of the image
ulSeconds	The second value of the image timestamp
ulMicroSeconds	The microsecond value of the image timestamp
pDataR	A buffer to hold the R data
pDataG	A buffer to hold the G data
pDataB	A buffer to hold the B data
triclopsInput	A TriclopsInput created by the FlyCapture image passed in. `Returns:` TriclopsErrorOk: The operation succeeded. TriclopsErrorFailed: The operation failed.

6.5.3. triclopsGetImage

Declaration
TriclopsError
triclopsGetImage(const TriclopsContext context,
TriclopsImageType imageType,
TriclopsCamera camera,
TriclopsImage* image)

Retrieves a specified type of image associated with the specified camera.

This function supplies the data to a TriclopsImage. The user is responsible for allocating the TriclopsImage structure. The data value of the structure will point to memory within the stereo kernel working space. Therefore, if a permanent copy of this image is desired, the user must copy it out.

Parameters

context	The context.
imageType	The image type requested.
camera	The camera that generated the requested image.
image	A TriclopsImage with the image data. `Returns:` TriclopsErrorOk: The operation succeeded.

	InvalidContext	The input context was invalid.
	InvalidCamera	The camera does not match a camera in this configuration.
	InvalidRequest	The image type does not match the camera or is not being generated by this system with the current context options.

See Also
triclopsSetImageBuffer()

6.5.4. triclopsGetImage16

Declaration
TriclopsError
triclopsGetImage16(const TriclopsContext context,
TriclopsImage16Type imageType,
TriclopsCamera camera,
TriclopsImage16* image)
Retrieves a specified 16-bit image associated with the specified camera.

This function performs the same action as triclopsGetImage(), except that it retrieves a 16-bit image type. Currently the only supported 16-bit image is the resultant disparity image from subpixel interpolation.

Parameters

context	The context.
imageType	The image type requested.
camera	The camera that generated the requested image.
image	A TriclospImage16 with the image data. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.
InvalidCamera	The camera does not match a camera in this configuration.
InvalidRequest	The image type does not match the camera or is not being generated by this system with the current context options.

See Also
triclopsGetImage()

6.5.5. triclopsSaveImage

Declaration
TriclopsError
triclopsSaveImage(TriclopsImage* image,
char* filename)
Saves an image to the specified filename. The file format currently supported is PGM format.

This function saves the input image to the requested file. Currently, this function will not detect if the file could not be opened, and will always show successful return.

Parameters

image	The TriclopsImage to be saved.
filename	The file name in which to save the image. Returns:

SystemError	The file could not be opened
TriclopsErrorOk	The operation succeeded.

6.5.6. triclopsSaveImage16

Declaration
TriclopsError
triclopsSaveImage16(TriclopsImage16* image,
char* filename)
Saves an image to the specified filename. The file format currently supported is PGM format.

This function saves the input image to the requested file. Currently, this function will not detect if the file could not be opened, and will always return successful.

Parameters

image	The TriclopsImage16 to be saved.
filename	The file name in which to save the image. Returns:

SystemError	The file could not be opened
TriclopsErrorOk	The operation succeeded.

See Also
triclopsSaveImage()

6.5.7. triclopsVersion

Declaration
const char*
triclopsVersion()
Returns a string with the Triclops library version.

Input:

Returns: char* - A string containing the version information for the context.

This function returns internally-handled memory. The caller should not free the returned pointer.

6.6. Stereo

6.6.1. triclopsGetDisparity

Declaration
TriclopsError
triclopsGetDisparity(const TriclopsContext context,
int* minDisparity,
int* maxDisparity)
Retrieves the disparity range from the given context.

Parameters

context	The context
minDisparity	A pointer to an integer that will store the current value.
maxDisparity	A pointer to an integer that will store the current value. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsGetDisparityOffset.

6.6.2. triclopsGetDisparityMapping

Declaration
TriclopsError
triclopsGetDisparityMapping(const TriclopsContext context,
unsigned char* minDisparity,
unsigned char* maxDisparity)
Retrieves the disparity range from the given context.

Parameters

context	The context.
minDisparity	The disparity range in the output disparity image minimum.
maxDisparity	The disparity range in the output disparity image maximum. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsSetDisparityMapping(), triclopsSetDisparity()

6.6.3. triclopsGetDisparityMappingOn

Declaration
TriclopsError
triclopsGetDisparityMappingOn(TriclopsContext context,
TriclopsBool* on)

Retrieves the current setting.

This function is used to extract the current disparity mapping setting.

Parameters

context	TriclopsContext for the operation.
on	A pointer that will contain the current value of this flag. Returns:

TriclopsErrorOk	Operation successful.
TriclopsErrorInvalidContext	Context is not a valid TriclopsContext.

See Also
triclopsSetDisparityMappingOn.

6.6.4. triclopsGetDisparityOffset

Declaration
TriclopsError
triclopsGetDisparityOffset(const TriclopsContext context,
int* nDisparityOffset)
Retrieves the disparity offset from the given context. Adding the disparity offset to a valid disparity value from the disparity image gives the true disparity value. The disparity offset is set automatically when the disparity range is set using triclopsSetDisparity, and is equal to MAX(0, maxDisparity-240). The disparity offset allows the disparity range, (which has a maximum extent of 240) to be shifted up away from zero, so that objects very close to the camera may be imaged.

Parameters

context	The context
nDisparityOffset	A pointer to an integer that will store the current value. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

6.6.5. triclopsGetDoStereo

Declaration
TriclopsError
triclopsGetDoStereo(const TriclopsContext context,
TriclopsBool* on)
Retrieves the state of the stereo processing.

Parameters

context	The context.
on	A pointer to a Boolean that will store the current setting. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsSetDoStereo(), triclopsSetStereoQuality(), triclopsGetStereoQuality()

6.6.6. triclopsGetEdgeCorrelation

Declaration
TriclopsError
triclopsGetEdgeCorrelation(const TriclopsContext context,
TriclopsBool* on)
Retrieves the state of the edge based correlation flag.

Parameters

context	The context.
on	A pointer to a Boolean that will contain the current setting. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsSetEdgeCorrelation()

6.6.7. triclopsGetEdgeMask

Declaration
TriclopsError
triclopsGetEdgeMask(const TriclopsContext context,
int* masksize)
Retrieves the edge detection mask size.

Parameters

context	The context.
masksize	A pointer to an integer that will contain the current mask size. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsSetEdgeMask()

6.6.8. triclopsGetMaxThreadCount

Declaration
TriclopsError
triclopsGetMaxThreadCount(TriclopsContext context,
int* maxThreadCount)
Retrieves the maximum number of threads used in multi-threaded calls

This function returns the maximum number of threads allowed in multuthreaded operations.

Parameters

context	The context.

See Also
triclopsSetMaxThreadCount(), triclopsStereo()

6.6.9. triclopsGetROIs

Declaration
TriclopsError
triclopsGetROIs(TriclopsContext context,
TriclopsROI** rois,
int* maxROIs)
Retrieves a handle to an array of Regions Of Interest.

This function returns a pointer to an array of region of interest (ROI) structures. The user may set requested region of interest boundaries in this array. The Regions Of Interest that will be valid must begin with the 0th element in the array and continue in a contiguous manner. After having set the Regions Of Interest, the user must call triclopsSetNumberOfROIs() to indicate to the stereo kernel how many ROIs he/she wishes processed. Currently, ROIs are only applied during the stereo processing, not during preprocessing.

Parameters

context	The context.
rois	A pointer to a 1D array of ROI structures.
maxROIs	The maximum number of ROIs allowed. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsSetNumberOfROIs(), triclopsStereo(), TriclopsROI

6.6.10. triclopsGetStereoMask

Declaration
TriclopsError
triclopsGetStereoMask(const TriclopsContext context,
int* size)
Retrieves the stereo correlation mask size.

Parameters

context	The context.
size	A pointer to an integer that will store the current value. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

6.6.11. triclopsGetStereoQuality

Declaration
TriclopsError
triclopsGetStereoQuality(TriclopsContext context,
TriclopsStereoQuality* quality)
Gets the quality of the stereo algorithm currently in use for stereo processing.

Parameters

context	The current TriclopsContext.
quality	The quality of the stereo algorithm currently in use. Returns:

TriclopsErrorOk	The operation succeeded.
TriclopsErrorInvalidContext	If the context is invalid.

See Also
triclopsSetStereoAlgQuality()

6.6.12. triclopsGetSubpixelInterpolation

Declaration
TriclopsError
triclopsGetSubpixelInterpolation(const TriclopsContext context,
TriclopsBool* on)
Retrieves the state of the subpixel interpolation feature.

Parameters

context	The context.
on	A pointer to a Boolean that will store the current setting. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The specified context was invalid.

See Also
triclopsSetSubpixelInterpolation(), triclopsSetStrictSubpixelValidation()

6.6.13. triclopsSetAnyStereoMask

Declaration
TriclopsError
triclopsSetAnyStereoMask(TriclopsContext context,
int size)
Allows the user to set stereomask to any size.

This function allows you to set a stereo correlation mask of any size. There is a danger that an internal counter in the stereo correlation engine may overflow if mask sizes over 15 are provided. Thus, the current limit for triclopsSetStereoMask is 15. However, in practice, much larger mask sizes can be used successfully. If an overflow results, it may not affect the stereo result, and it will generally only happen for pathological cases.

Therefore, this function is provided as an 'experimental' function to allow users to set any mask size they wish.

Parameters

context	TriclopsContext for the operation.
size	The size for a new stereo correlation mask. Returns:

TriclopsErrorOk	Operation successful.
TriclopsErrorInvalidContext	Context is not a valid TriclopsContext.
TriclopsErrorInvalidSetting	A value of less than 1

See Also
triclopsSetStereoMask()

6.6.14. triclopsSetDisparity

Declaration
TriclopsError
triclopsSetDisparity(TriclopsContext context,
int minDisparity,
int maxDisparity)
Sets the disparity range for stereo processing. Currently, the range must be such that (maxDisparity-minDisparity<=240) and (maxDisparity<=1024). Note that a side-effect of setting maxDisparity > 240 is that the disparity offset is set to maxDisparity-240.

Parameters

context	The context.
minDisparity	The disparity range minimum.
maxDisparity	The disparity range maximum. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsGetDisparityOffset.

6.6.15. triclopsSetDisparityMapping

Declaration
TriclopsError
triclopsSetDisparityMapping(TriclopsContext context,
unsigned char minDisparity,
unsigned char maxDisparity)
Sets the disparity range for stereo processing.

This function sets the disparity mapping values. The disparity mapping values control what range of pixels values appear in the output disparity image. The true disparity ranges between the minimum and maximum values set with triclopsSetDisparity(). The output image has its pixel values linearly scaled from minimum => maximum disparity to

minimum => maximum disparity mapping. This is primarily used when one wishes to use a screen display buffer as the disparity image buffer. Note: it is advisable to set the disparity mapping to the exact values of the input disparities if one is not using the screen buffer display feature.

Parameters

context	The context.
minDisparity	The disparity range in the output disparity image minimum. The range is between 0 and 255.
maxDisparity	The disparity range in the output disparity image maximum. The range is between 0 and 255. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsGetDisparityMapping(), triclopsSetDisparity(), triclopsRCD8ToXYZ(), triclopsRCD16ToXYZ(), triclopsRCDMappedToXYZ()

6.6.16. triclopsSetDisparityMappingOn

Declaration
TriclopsError
triclopsSetDisparityMappingOn(TriclopsContext context,
TriclopsBool on)
Enables or disables disparity mapping.

This function is used to enable or disable disparity mapping. See the comments section on this subject in Chapter 4 in the Triclops manual for why disparity mapping can cause trouble.

Parameters

context	TriclopsContext for the operation.
on	A Boolean flag indicating if the mapping should be on or off. Returns:

TriclopsErrorOk	Operation successful.
TriclopsErrorInvalidContext	Context is not a valid TriclopsContext.

6.6.17. triclopsSetDoStereo

Declaration
TriclopsError
triclopsSetDoStereo(TriclopsContext context,
TriclopsBool on)
Turns stereo processing on or off.

Parameters

context	The context.
on	A Boolean indicating whether stereo processing should be turned on or off.

	Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid.

See Also
triclopsGetDoStereo(), triclopsSetStereoQuality(), triclopsGetStereoQuality()

6.6.18. triclopsSetEdgeCorrelation

Declaration
TriclopsError
triclopsSetEdgeCorrelation(TriclopsContext context,
TriclopsBool on)
Turns edge based correlation on or off.

Edge based correlation is required for texture and uniqueness validation to be used.

Parameters

context	The context.
on	A Boolean value indicating whether correlation should be turned on or off. Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid.

See Also
triclopsGetEdgeCorrelation(), triclopsSetTextureValidation(),
triclopsSetUniquenessValidation()

6.6.19. triclopsSetEdgeMask

Declaration
TriclopsError
triclopsSetEdgeMask(TriclopsContext context,
int masksize)
Sets the edge detection mask size.

Parameters

context	The context.
masksize	The new mask size, which is valid between 3 and 11. Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid.

See Also
triclopsGetEdgeMask()

6.6.20. triclopsSetMaxThreadCount

Declaration

TriclopsError
triclopsSetMaxThreadCount(TriclopsContext context,
int maxThreadCount)
Sets the maximum number of threads to use in multi-threaded operations

This function indicates to the stereo kernel at what number to cap the number of threads used in multi-threaded operations. If not set, the kernel will use a default number based on the architecture of the machine and the expected sppedup.

Parameters

context	The context. nrois – maximum number of threads (minimum 1) Returns: TriclopsErrorOk – The operation succeeded. InvalidContext – The input context was invalid. InvalidSetting – Invalid number passed in (less then 1). InvalidRequest – the kernel is busy executing in the background

See Also
triclopsGetMaxThreadCount(), triclopsStereo()

6.6.21. triclopsSetNumberOfROIs

Declaration
TriclopsError
triclopsSetNumberOfROIs(TriclopsContext context,
int nrois)
Sets the number of Regions Of Interest the user currently wants active.

This function indicates to the stereo kernel how many ROIs in the ROI array will be processed. Before calling this function, the user must first call triclopsGetROIs() and set up the ROIs he/she intends to use. If nrois is set to 0, the entire image will be processed. This is the default setting.

Parameters

context	The context.
nrois	Number of ROIs. Returns: TriclopsErrorOk – The operation succeeded. InvalidContext – The input context was invalid. InvalidSetting – Too many ROIs.

See Also
triclopsGetROIs()

6.6.22. triclopsSetStereoMask

Declaration

TriclopsError
triclopsSetStereoMask(TriclopsContext context,
int masksize)
Set the stereo correlation mask size.

Parameters

context	The context.
masksize	The new correlation mask size, which is valid between 1 and 15. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

6.6.23. triclopsSetStereoQuality

Declaration
TriclopsError
triclopsSetStereoQuality(TriclopsContext context,
TriclopsStereoQuality quality)
Sets the quality of the stereo algorithm to use during stereo processing. Higher-quality algorithms will generally require more processing time, but give more accurate and/or more complete results. Currently available are TriStereoQlty_STANDARD, our classic stereo algorithm, and TriStereoQlty_ENHANCED, which provides a more reliable sub-pixel disparity estimate.

Parameters

context	The current TriclopsContext.
quality	The desired quality of the stereo algorithm to use Returns:

TriclopsErrorOk	The operation succeeded.
TriclopsErrorInvalidContext	If the context is invalid.

See Also
triclopsGetStereoAlgQuality()

6.6.24. triclopsSetSubpixelInterpolation

Declaration
TriclopsError
triclopsSetSubpixelInterpolation(TriclopsContext context,
TriclopsBool on)
Turns subpixel interpolation stereo improvements on or off.

Parameters

context	The context.
on	A Boolean indicating whether it should be on or off. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsGetSubpixelInterpolation(), triclopsSetStrictSubpixelValidation()

6.6.25. triclopsStereo

Declaration
TriclopsError
triclopsStereo(TriclopsContext context)
Does stereo processing, validation, and subpixel interpolation, as specified by parameters.

This function performs the stereo processing and validation, and generates the internal image TriImg_DISPARITY.

Parameters

context	The context. Returns:

TriclopsErrorOk	The operation succeeded.
BadOptions	There are some contradictory options set.
NotImplemented	The context camera configuration is not supported.
InvalidROI	ROI has negative dimensions or illegal starting position.
InvalidRequest	triclopsStereo called before triclopsPreprocess was called.

Remarks
An illegal starting position is a upper/left corner that is closer to the right/bottom edge of the image than the stereo mask size.

See Also
triclopsPreprocessing(), triclopsSetDoStereo(), triclopsEdgeCorrelation(), triclopsSetTextureValidation(), triclopsSetUniquenessValidation(), triclopsGetROIs(), triclopsSetSubpixelInterpolation()

6.7. Configuration

6.7.1. triclopsGetBaseline

Declaration
TriclopsError
triclopsGetBaseline(TriclopsContext context,
float* base)
Retrieves the baseline of the cameras.

This function retrieves the baseline of the cameras in meters. The stereo context must have already been read.

Parameters

context	The context.
base	The baseline in meters. Returns:

	TriclopsErrorOk	The operation succeeded.
	InvalidContext	The input context was invalid.

See Also
triclopsGetDefaultContextFromFile()

6.7.2. triclopsGetCameraConfiguration

Declaration
TriclopsError
triclopsGetCameraConfiguration(const TriclopsContext context,
TriclopsCameraConfiguration* config)
Retrieves the current configuration of the stereo camera. This configuration is the configuration that specifies the stereo algorithm used. For example, 2CAM_HORIZONTAL configuration can be used to do 2 camera stereo on a 3 camera device.

Parameters

context	The context.
config	A pointer that will hold the result. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
TriclopsCameraConfiguration, TriclopsSetCameraConfiguration(), TriclopsGetDeviceConfiguration()

6.7.3. triclopsGetDeviceConfiguration

Declaration
TriclopsError
triclopsGetDeviceConfiguration(const TriclopsContext context,
TriclopsCameraConfiguration* config)
This function returns the physical configuration of the stereo device. This allows the user to determine what algorithms they have available on the current device.

Parameters

context	The context.
config	The physical CameraConfiguration. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
TriclopsCameraConfiguration, TriclopsGetCameraConfiguration(), TriclopsSetCameraConfiguration()

6.7.4. triclopsGetFocalLength

Declaration
TriclopsError
triclopsGetFocalLength(const TriclopsContext context,
float* focallength)
Retrieves the focal length of the cameras.

This function returns the focal length of the system. The focal length is in 'pixels' for the current selected output resolution. All cameras' rectified images have the same focal length. The default stereo context must have been read before this call can be made.

Parameters

context	The context.
focallength	The focal length in pixels. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsGetDefaultContextFromFile(), triclopsSetResolution()

6.7.5. triclopsGetImageCenter

Declaration
TriclopsError
triclopsGetImageCenter(TriclopsContext context,
float* centerRow,
float* centerCol)
Returns the optical center for pinhole calculations.

It is important that the context already has the resolution set. If triclopsSetResolution is not set, the returned value cannot be used for calculations. This image center can be used as the position in the image plane of the optical center for pinhole camera calculations.

Parameters

context	TriclopsContext for the operation.
centerRow	A pointer that will contain the row position of the image center for the current resolution.
centerCol	A pointer that will contain the column position of the image center for the current resolution. Returns:

TriclopsErrorOk	Operation successful.
TriclopsErrorInvalidContext	Context is not a valid TriclopsContext.

See Also
triclopsSetResolution()

6.7.6. triclopsGetSerialNumber

Declaration
TriclopsError
triclopsGetSerialNumber(const TriclopsContext context,
int* serialNumber)
This function returns the serial number of the stereo camera product associated with the given TriclopsContext.

Parameters

context	The context to extract the serial number from.
serialNumber	The serial number of the stereo camera product associated with the given context. Returns: TriclopsErrorOk — Upon the successful completion of the operation.

6.7.7. triclopsSetCameraConfiguration

Declaration
TriclopsError
triclopsSetCameraConfiguration(const TriclopsContext context,
TriclopsCameraConfiguration config)
Sets the configuration of the cameras. This configuration determines the configuration for the stereo algorithm. For example, a three camera stereo device may be set into "2 camera horizontal" mode for faster stereo processing.

Parameters

context	The context.
config	The new CameraConfiguration. Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid. InvalidParameter — 'config' is not a valid camera configuration

See Also
TriclopsCameraConfiguration, TriclopsGetCameraConfiguration(), TriclopsGetDeviceConfiguration()

6.8. 3D

6.8.1. triclopsCreateImage3d

Declaration
TriclopsError
triclopsCreateImage3d(TriclopsContext context,
TriclopsImage3d** ppimage)

Allocates a TriclopsImage3d to the correct size as specified by the resolution of the TriclopsContext

Parameters

context	The current TriclopsContext.
ppimage	Pointer to the address of the TriclopsImage3d structure to allocate memory for. Returns:

TriclopsErrorOk	The operation succeeded.
TriclopsErrorSystemError	If there was a problem allocating the memory.

See Also
triclopsDestroyImage3d(), triclopsExtractImage3d(), triclopsExtractWorldImage3d()

6.8.2. triclopsDestroyImage3d

Declaration
void
triclopsDestroyImage3d(TriclopsImage3d** ppimage)
Deallocates a TriclopsImage3d allocated by triclopsCreateImage3d()

Parameters

ppimage	Pointer to the address of the TriclopsImage3d structure to destroy.

See Also
triclopsCreateImage3d(), triclopsExtractImage3d(), triclopsExtractWorldImage3d()

6.8.3. triclopsExtractImage3d

Declaration
TriclopsError
triclopsExtractImage3d(TriclopsContext context,
TriclopsImage3d* pimage)
Creates a 3D image given the current disparity result of a TriclopsContext

Parameters

context	The current TriclopsContext.
pimage	Pointer to the TriclopsImage3d structure. Returns:

TriclopsErrorOk	The operation succeeded.
TriclopsErrorInvalidContext	If the context is invalid.
TriclopsErrorInvalidParameter	If there is a geometry mismatch between the context and the TriclopsImage3d.
TriclopsErrorInvalidRequest	If the (subpixel) disparity image does not match the current resolution.

Remarks
Invalid points will be assigned the value of (0,0,0) in the returned image.

See Also
triclopsDestroyImage3d(), triclopsCreateImage3d(), triclopsExtractWorldImage3d()

6.8.4. triclopsExtractWorldImage3d

Declaration
TriclopsError
triclopsExtractWorldImage3d(TriclopsContext context,
TriclopsImage3d* pimage)
Creates a 3D image given the current disparity result of a TriclopsContext that is transformed to the world coordinate system

Parameters

context	The current TriclopsContext.
pimage	Pointer to the TriclopsImage3d structure. Returns:

TriclopsErrorOk	The operation succeeded.
TriclopsErrorInvalidContext	If the context is invalid.
TriclopsErrorInvalidParameter	If there is a geometry mismatch between the context and the TriclopsImage3d.
TriclopsErrorInvalidRequest	If the (subpixel) disparity image does not match the current resolution.

Remarks
Invalid points will be assigned the value of (0,0,0) in the returned image.

See Also
triclopsDestroyImage3d(), triclopsCreateImage3d(), triclopsExtractImage3d()

6.8.5. triclopsGetTransformFromFile

Declaration
TriclopsError
triclopsGetTransformFromFile(char* fileName,
TriclopsTransform* transform)
Loads the contents of a TriclopsTransform from the input file.

This function fills in the provided TriclopsTransform structure based on the contents read from the specified file. If the specified file is not found, contains invalid fields, the value of CorruptTransformFile will be returned.

Parameters

fileName	name of the file from which to load the transform
transform	the 4x4 homogeneous transform Returns:

TriclopsErrorOk	The Operation succeeded.
CorruptTransformFile	either the file is not found or it contains invalid fields.

See Also
triclopsRCDToWorldXYZ(), triclopsRCDFloatToWorldXYZ(), triclopsRCD8ToWorldXYZ(), triclopsRCD16ToWorldXYZ(),

triclopsWorldXYZToRCD(), triclopsSetTriclopsToWorldTransform(),
triclopsGetTriclopsToWorldTransform(), triclopsWriteTransformToFile

6.8.6. triclopsGetTriclopsToWorldTransform

Declaration
TriclopsError
triclopsGetTriclopsToWorldTransform(TriclopsContext context,
TriclopsTransform* transform)
Gets the Triclops to World transform.

This function fills in the provided TriclopsTransform structure with the current contents of the Triclops to World transform for this TriclopsContext

Parameters

context	the TriclopsContext
transform	the 4x4 homogeneous transform Returns:

TriclopsErrorOk	The Operation succeeded.
TriclopsErrorInvalidContext	Context is not valid TriclopsContext.

See Also
triclopsRCDToWorldXYZ(), triclopsRCDFloatToWorldXYZ(),
triclopsRCD8ToWorldXYZ(), triclopsRCD16ToWorldXYZ(),
triclopsWorldXYZToRCD(), triclopsSetTriclopsToWorldTransform()
triclopsGetTransformFromFile(), triclopsWriteTransformToFile()

6.8.7. triclopsRCD16ToWorldXYZ

Declaration
TriclopsError
triclopsRCD16ToWorldXYZ(TriclopsContext context,
int row,
int col,
unsigned short disp,
float* x,
float* y,
float* z)
Converts image coordinates and a 16-bit disparity into a world 3D point.

This function takes a 16-bit disparity value and converts it to XYZ coordinates in world coordinates. The world coordinates are determined by transforming the point from the Triclops coordinate system to the world coordinate system based on the TriclopsContext transform.

Parameters

context	The stereo context.
row	The row of the input pixel.
col	The column of the input pixel.

disp	The disparity value of the input pixel.
x	The x coordinate of the corresponding 3D point
y	The y coordinate of the corresponding 3D point
z	The z coordinate of the corresponding 3D point Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

Remarks
It is up to the user to supply valid disparity values

Disparity offset SHOULD NOT be applied to input disp values. In this way, users can step through a disparity image calling this function and passing image disparity values straight in with no changes.

See Also
triclopsRCDFloatToXYZ(), triclopsRCDMappedToXYZ(), triclopsRCD8ToXYZ(), triclopsRCD16ToXYZ(), triclopsSetDisparity(), triclopsSetDisparityMapping() triclopsRCDToWorldXYZ(), triclopsRCDFloatToWorldXYZ(), triclopsRCD8ToWorldXYZ(), triclopsSetTriclopsToWorldTransform(), triclopsGetTriclopsToWorldTransform() triclopsGetDisparityOffset()

6.8.8. triclopsRCD16ToXYZ

Declaration
TriclopsError
triclopsRCD16ToXYZ(TriclopsContext context,
int row,
int col,
unsigned short disp,
float* x,
float* y,
float* z)
Converts image coordinates and a 16-bit disparity into true 3D points.

When using this function, you should ensure that the values for the Triclops disparity mapping feature are the same as the disparity range.

Parameters

context	The stereo context.
row	The row of the input pixel.
col	The column of the input pixel.
disp	The disparity value of the input pixel.
x	The x coordinate of the point represented by the input row column disparity in the camera coordinate system.
y	The y coordinate of the point represented by the input row column disparity in the camera coordinate system.
z	The z coordinate of the point represented by the input row column disparity in the camera coordinate system.

	Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid.

Remarks
It is up to the user to supply valid pixel locations. Pixels that have been invalidated may give negative results.

Disparity offset SHOULD NOT be applied to input disp values. In this way, users can step through a disparity image calling this function and passing image disparity values straight in with no changes.

See Also
triclopsRCDFloatToXYZ(), triclopsRCDMappedToXYZ(), triclopsRCD8ToXYZ(), triclopsSetDisparity(), triclopsSetDisparityMapping() triclopsGetDisparityOffset()

6.8.9. triclopsRCD8ToWorldXYZ

Declaration
```
TriclopsError
triclopsRCD8ToWorldXYZ( TriclopsContext context,
                        int row,
                        int col,
                        unsigned char disp,
                        float* x,
                        float* y,
                        float* z )
```
Converts image coordinates and an 8-bit disparity into a world 3D point.

This function takes an 8-bit disparity value and converts it to XYZ coordinates in world coordinates. The world coordinates are determined by transforming the point from the Triclops coordinate system to the world coordinate system based on the TriclopsContext transform.

Parameters

context	The stereo context.
row	The row of the input pixel.
col	The column of the input pixel.
disp	The disparity value of the input pixel.
x	The x coordinate of the corresponding 3D point
y	The y coordinate of the corresponding 3D point
z	The z coordinate of the corresponding 3D point Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid.

Remarks
It is up to the user to supply valid disparity values

Disparity offset SHOULD NOT be applied to input disp values. In this way, users can step through a disparity image calling this function and passing image disparity values straight in with no changes.

See Also
triclopsRCDFloatToXYZ(), triclopsRCDMappedToXYZ(), triclopsRCD16ToXYZ(), triclopsSetDisparity(), triclopsSetDisparityMapping() triclopsRCDToWorldXYZ(), triclopsRCDFloatToWorldXYZ(), triclopsRCD16ToWorldXYZ(), triclopsSetTriclopsToWorldTransform(), triclopsGetTriclopsToWorldTransform() triclopsGetDisparityOffset()

6.8.10. triclopsRCD8ToXYZ

Declaration
TriclopsError
triclopsRCD8ToXYZ(TriclopsContext context,
int row,
int col,
unsigned char disp,
float* x,
float* y,
float* z)
Converts image coordinates and an *-bit disparity into true 3D points.

When using this function, you should ensure that the values for the Triclops disparity mapping feature are the same as the disparity range.

Parameters

context	The stereo context.
row	The row of the input pixel.
col	The column of the input pixel.
disp	The disparity value of the input pixel.
x	The x coordinate of the point represented by the input row column disparity in the camera coordinate system.
y	The y coordinate of the point represented by the input row column disparity in the camera coordinate system.
z	The z coordinate of the point represented by the input row column disparity in the camera coordinate system. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

Remarks
It is up to the user to supply valid pixel locations. Pixels that have been invalidated may give negative results.

Disparity offset SHOULD NOT be applied to input disp values. In this way, users can step through a disparity image calling this function and passing image disparity values straight in with no changes.

See Also
triclopsRCDFloatToXYZ(), triclopsRCDMappedToXYZ(), triclopsRCD16ToXYZ(), triclopsSetDisparity(), triclopsSetDisparityMapping(), triclopsGetDisparityOffset()

6.8.11. triclopsRCDFloatToWorldXYZ

Declaration
TriclopsError
triclopsRCDFloatToWorldXYZ(TriclopsContext context,
float row,
float col,
float disp,
float* x,
float* y,
float* z)
Converts an image location and a floating-point disparity value into a world 3D point.

This function takes a floating-point disparity value and converts it to XYZ coordinates in world coordinates.

Parameters

context	The stereo context.
row	The row of the input pixel.
col	The column of the input pixel.
disp	The disparity value of the input pixel.
x	The x coordinate of the corresponding 3D point
y	The y coordinate of the corresponding 3D point
z	The z coordinate of the corresponding 3D point Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

Remarks
The world coordinates are determined by transforming the point from the Triclops coordinate system to the world coordinate system based on the TriclopsContext transform.

It is up to the user to supply valid disparity values.

Disparity offset MUST be applied to input disp values, if offset != 0, in order to produce valid XYZ values.

See Also
triclopsRCDMappedToXYZ(), triclopsRCD8ToXYZ(), triclopsRCD16ToXYZ() triclopsRCDToWorldXYZ(), triclopsRCD8ToWorldXYZ(), triclopsRCD16ToWorldXYZ(), triclopsSetTriclopsToWorldTransform(), triclopsGetTriclopsToWorldTransform() triclopsGetDisparityOffset()

6.8.12. triclopsRCDFloatToXYZ

Declaration
TriclopsError
triclopsRCDFloatToXYZ(TriclopsContext context,
float row,
float col,
float disp,
float* x,
float* y,
float* z)
Converts image coordinates and a floating-point disparity value into true 3D points.

This function takes a floating-point disparity value and converts it to XYZ coordinates.

Parameters

context	The stereo context.
row	The row of the input pixel.
col	The column of the input pixel.
disp	The disparity value of the input pixel.
x	The x coordinate of the point represented by the input row column disparity in the camera coordinate system.
y	The y coordinate of the point represented by the input row column disparity in the camera coordinate system.
z	The z coordinate of the point represented by the input row column disparity in the camera coordinate system. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

Remarks
It is up to the user to supply valid pixel locations. Pixels which have been invalidated may give negative results.

Disparity offset MUST be applied to input disp values, if offset != 0, in order to produce valid XYZ values.

See Also
triclopsRCDMappedToXYZ(), triclopsRCD8ToXYZ(), triclopsRCD16ToXYZ(), triclopsGetDisparityOffset()

6.8.13. triclopsRCDMappedToWorldXYZ

Declaration
TriclopsError
triclopsRCDMappedToWorldXYZ(TriclopsContext context,
int row,
int col,

unsigned char disp,
float* x,
float* y,
float* z)

Converts image coordinates with disparity values that have been mapped using the disparity mapping function into world 3D points.

This function takes disparity values that have been scaled by the disparity mapping feature and transforms them into a "world" coordinate system based on the transform recorded in the TriclopsContext.

Parameters

context	The stereo context.
row	The row of the input pixel.
col	The column of the input pixel.
disp	The disparity value of the input pixel.
x	The x coordinate of the point represented by the input row column disparity in the camera coordinate system.
y	The y coordinate of the point represented by the input row column disparity in the camera coordinate system.
z	The z coordinate of the point represented by the input row column disparity in the camera coordinate system. Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid.

Remarks

If you have set "Disparity Mapping" on you should use this function. However, it is less efficient than the other XYZ conversion functions and may have some round-off errors. It is preferable to set the disparity mapping off.

It is up to the user to supply valid disparity values, invalid disparity values (as taken from an invalid pixel in the a disparity image) may give negative results.

Disparity offset SHOULD NOT be applied to input disp values. In this way, users can step through a disparity image calling this function and passing image disparity values straight in with no changes.

See Also

triclopsRCDFloatToXYZ(), triclopsRCD8ToXYZ(), triclopsRCD16ToXYZ()
triclopsRCDToWorldXYZ(), triclopsRCDFloatToWorldXYZ(),
triclopsRCD8ToWorldXYZ(), triclopsRCD16ToWorldXYZ(),
triclopsSetTriclopsToWorldTransform(), triclopsGetTriclopsToWorldTransform()
triclopsGetDisparityOffset()

6.8.14. triclopsRCDMappedToXYZ

Declaration

TriclopsError
triclopsRCDMappedToXYZ(TriclopsContext context,
int row,
int col,
unsigned char disp,
float* x,
float* y,
float* z)

Converts image coordinates with disparity values that have been mapped using the disparity mapping function into true 3D points.

This function takes disparity values that have been scaled by the disparity mapping feature.

Parameters

context	The stereo context.
row	The row of the input pixel.
col	The column of the input pixel.
disp	The disparity value of the input pixel.
x	The x coordinate of the point represented by the input row column disparity in the camera coordinate system.
y	The y coordinate of the point represented by the input row column disparity in the camera coordinate system.
z	The z coordinate of the point represented by the input row column disparity in the camera coordinate system. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

Remarks

If you do not have the disparity mapping values set to the same as the disparity values, you should use this function. However, it is less efficient than the other XYZ conversion functions and may have some round-off errors. It is preferable to set the disparity mapping range to the same as the disparity range and use one of the other conversion functions.

It is up to the user to supply valid pixel locations. Pixels that have been invalidated may give negative results.

Disparity offset SHOULD NOT be applied to input disp values. In this way, users can step through a disparity image calling this function and passing image disparity values straight in with no changes.

See Also

triclopsRCDFloatToXYZ(), triclopsRCD8ToXYZ(), triclopsRCD16ToXYZ()

6.8.15. triclopsRCDToWorldXYZ

Declaration

TriclopsError
triclopsRCDToWorldXYZ(TriclopsContext context,
float row,
float col,
float disp,
float* x,
float* y,
float* z)

Converts image coordinates and disparity values to world 3D points.

This function takes a pixel location and matching disparity value and calculates the 3D position that this combination represents. The position is calculated in the "world" coordinate system. That is to say, the position is calculated in the Triclops coordinate system and then transformed by the TriclopsContext transform to a new coordinate system.

Parameters

context	The stereo context.
row	The row of the input pixel.
col	The column of the input pixel.
disp	The disparity value of the input pixel.
x	The x coordinate of the corresponding 3D point in the world coordinate system
y	The y coordinate of the corresponding 3D point in the world coordinate system
z	The z coordinate of the corresponding 3D point in the world coordinate system Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

Remarks
It is up to the user to supply valid disparity values. Values taken from invalid pixels in the disparity image give negative results.

Disparity offset SHOULD NOT be applied to input disp values. In this way, users can step through a disparity image calling this function and passing image disparity values straight in with no changes.

See Also
triclopsRCDFloatToXYZ(), triclopsRCD8ToXYZ(), triclopsRCD16ToXYZ(), triclopsRCDFloatToWorldXYZ(), triclopsRCD8ToWorldXYZ(), triclopsRCD16ToWorldXYZ(), triclopsSetTriclopsToWorldTransform(), triclopsGetTriclopsToWorldTransform(), triclopsGetDisparityOffset()

6.8.16. triclopsRCDToXYZ

Declaration
TriclopsError
triclopsRCDToXYZ(TriclopsContext context,
float row,
float col,

float disp,
float* x,
float* y,
float* z)

Converts image coordinates with disparity values that have been mapped using the disparity mapping function into true 3D points.

This function takes disparity values that have been scaled by the disparity mapping feature.

Parameters

context	The stereo context.
row	The row of the input pixel.
col	The column of the input pixel.
disp	The disparity value of the input pixel.
x	The x coordinate of the point represented by the input row column disparity in the camera coordinate system.
y	The y coordinate of the point represented by the input row column disparity in the camera coordinate system.
z	The z coordinate of the point represented by the input row column disparity in the camera coordinate system. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

Remarks

If you do not have the disparity mapping values set to the same as the disparity values, you should use this function. However, it is less efficient than the other XYZ conversion functions and may have some round-off errors. It is preferable to set the disparity mapping range to the same as the disparity range and use one of the other conversion functions.

It is up to the user to supply valid pixel locations. Pixels that have been invalidated may give negative results.

Disparity offset SHOULD NOT be applied to input disp values. In this way, users can step through a disparity image calling this function and passing image disparity values straight in with no changes.

See Also

triclopsRCDFloatToXYZ(), triclopsRCD8ToXYZ(), triclopsRCD16ToXYZ()

6.8.17. triclopsSetTriclopsToWorldTransform

Declaration

TriclopsError
triclopsSetTriclopsToWorldTransform(TriclopsContext context,
TriclopsTransform transform)

Sets the Triclops to World transform.

This function sets the internal TriclopsContext transform to match the provided transform.

Parameters

context	the TriclopsContext
transform	the 4x4 homogeneous transform

Remarks
There are several things to note:

1. The transform is the Triclops-to-World transform. ie: when this transform is applied to a Triclops point, it will change it to a world point.

2. The transform must be of the approved format. This means it has a 3x3 rotational component that is orthonormal and the bottom row must be of format (0 0 0 n). This function will try to normalize the rotational component and clean up the bottom row of the matrix. One can verify whether modifications to the transform were necessary by obtaining the current transform using triclopsGetTriclopsToWorldTransform() and comparing.

See Also
triclopsRCDToWorldXYZ(), triclopsRCDFloatToWorldXYZ(), triclopsRCD8ToWorldXYZ(), triclopsRCD16ToWorldXYZ(), triclopsWorldXYZToRCD(), triclopsGetTriclopsToWorldTransform() triclopsGetTransformFromFile(), triclopsWriteTransformToFile()

6.8.18. triclopsWorldXYZToRCD

Declaration
TriclopsError
triclopsWorldXYZToRCD(TriclopsContext context,
float x,
float y,
float z,
float* row,
float* col,
float* disp)
Converts world 3D points into image coordinates.

This function takes as input the XYZ position of a point in the World coordinate system, moves the point to the Triclops coordinate system (as described by the TriclopsContext transform), and determines what row, column, and disparity value would result from the resulting point.

Parameters

context	TriclopsContext set up for desired resolution.
x	X value of a point in the World coordinate system.
y	Y value of a point in the World coordinate system.
z	Z value of a point in the World coordinate system.
row	The row in a disparity image.

col	The column in a disparity image.
disp	The disparity value that would match the point specified in XYZ. Returns:

TriclopsErrorOk	The Operation succeeded.
TriclopsErrorInvalidContext	Context is not valid TriclopsContext.
TriclopsErrorInvalidRequest	An impossible XYZ value has been provided (ie: negative Z).

See Also
triclopsRCDFloatToXYZ(), triclopsRCDMappedToXYZ(), triclopsRCD8ToXYZ(), triclopsSetDisparity(), triclopsSetDisparityMapping() triclopsRCDToWorldXYZ(), triclopsRCDFloatToWorldXYZ(), triclopsRCD8ToWorldXYZ(), triclopsRCD16ToWorldXYZ(), triclopsXYZToRCD(), triclopsSetTriclopsToWorldTransform(), triclopsGetTriclopsToWorldTransform()

6.8.19. triclopsWriteTransformToFile

Declaration
TriclopsError
triclopsWriteTransformToFile(char* fileName,
TriclopsTransform* transform)
Saves the contents of a TriclopsTransform to the output file.

This function saves the contents of the specified transform to an external file.

Parameters

fileName	name of the file to which to save the transform
transform	the 4x4 homogeneous transform to save

See Also
triclopsRCDToWorldXYZ(), triclopsRCDFloatToWorldXYZ(), triclopsRCD8ToWorldXYZ(), triclopsRCD16ToWorldXYZ(), triclopsWorldXYZToRCD(), triclopsSetTriclopsToWorldTransform(), triclopsGetTriclopsToWorldTransform(), triclopsGetTransformFromFile

6.8.20. triclopsXYZToRCD

Declaration
TriclopsError
triclopsXYZToRCD(TriclopsContext context,
float x,
float y,
float z,
float* row,
float* col,
float* disp)
Converts true 3D points into image coordinates.

This function takes as input the XYZ position of a point in the Triclops coordinate system, and determines what row, column, and disparity value would result from a sensed point at XYZ.

Parameters

context	TriclopsContext set up for desired resolution.
x	X value of a point in the Triclops coordinate system.
y	Y value of a point in the Triclops coordinate system.
z	Z value of a point in the Triclops coordinate system.
row	The row in a disparity image.
col	The column in a disparity image.
disp	The disparity value that would match the point specified in XYZ. Returns:

TriclopsErrorOk	The Operation succeeded.
TriclopsErrorInvalidContext	Context is not valid TriclopsContext.
TriclopsErrorInvalidRequest	An impossible XYZ value has been provided (ie: negative Z).

6.9. Image Buffer Operations

6.9.1. triclopsSetColorImageBuffer

Declaration
TriclopsError
triclopsSetColorImageBuffer(TriclopsContext context,
TriclopsCamera nCamera,
unsigned char* red,
unsigned char* green,
unsigned char* blue)

Allows the user to set separate buffers to which individual bands of the processed color image are written to. In this case, the "processed" image means the rectified image that is rectified when triclopsRectifyColorImage() is called.

Parameters

context	The TriclopsContext to set the buffer for.
nCamera	The camera buffer to set.
red	A pointer to the red buffer.
green	A pointer to the green buffer.
blue	A pointer to the blue buffer. Returns:

TriclopsErrorOk	Upon the successful completion of the operation.

See Also
triclopsRectifyColorImage()

6.9.2. triclopsSetImage16Buffer

Declaration
TriclopsError
triclopsSetImage16Buffer(TriclopsContext context,
unsigned short* buffer,
TriclopsImage16Type imageType,
TriclopsCamera camera)
This function allows the user to set the location to which 16bit (generally subpixel) depth images are written to once they are processed.

Parameters

context	The TriclopsContext to set the buffer in.
buffer	A pointer to the buffer.
imageType	The type of image to be written to the buffer.
camera	The camera to write from. Returns: TriclopsErrorOk: Upon the successful completion of the operation.

6.9.3. triclopsSetImageBuffer

Declaration
TriclopsError
triclopsSetImageBuffer(TriclopsContext context,
unsigned char* buffer,
TriclopsImageType imageType,
TriclopsCamera camera)
Sets the internal image buffer for the specified camera and image type to be the buffer supplied by the user.

This function allows the user to specify directly what memory he/she wishes the output images to be deposited into. This memory will be used by the stereo kernel as working space. This has the advantage of saving a copy for tasks such as displaying to the screen. The user may simply set the output image buffer to his/her display buffer. However, since this memory will be used by the stereo kernel as working space, the contents of the buffer may change with each call of triclopsPreprocess() or triclopsStereo(). If the results are to be saved, it is the user's responsibility to do so. In addition, the user is responsible to allocate sufficient memory for the buffer, and to de-allocate the buffer after it is no longer needed. Before de-allocating the buffer, the user should call triclopsUnsetImageBuffer(). If the user requests an invalid image, such as the disparity image, when the context stereo flag is set to false, or the edge image when edge correlation is set to false, an error of invalid request will be returned. Disparity images are always associated with the reference camera.

Parameters

context	The context.
buffer	A user allocated buffer of sufficient size.

imageType	The image type.
camera	The camera. Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid. InvalidCamera — The input camera is invalid for this camera configuration, or is not associated with the requested image type. InvalidRequest — The image is not set to be generated by the stereo kernel.

See Also
triclopsGetImage(), triclopsUnsetImageBuffer()

6.9.4. triclopsSetPackedColorImageBuffer

Declaration
TriclopsError
triclopsSetPackedColorImageBuffer(TriclopsContext context,
TriclopsCamera nCamera,
TriclopsPackedColorPixel* buffer)
Allows the user to set the buffer to which the processed color image is written to.

Parameters

context	The TriclopsContext to set the buffer for.
nCamera	The camera buffer to set.
buffer	A pointer to a buffer of TriclopsPackedColorPixels. Returns: TriclopsErrorOk — Upon the successful completion of the operation.

6.9.5. triclopsUnsetColorImageBuffer

Declaration
TriclopsError
triclopsUnsetColorImageBuffer(TriclopsContext context,
TriclopsCamera camera)
This releases the user specified internal color image buffer for the specified camera. The next time this buffer is required by the system, it will allocate a new one for internal use.

If the user has already called triclopsSetColorImageBuffer for a particular camera, the stereo kernel will be using that buffer when rectifying a color image. If the user no longer wants to have the supplied buffer used by the stereo kernel, he/she may use this function to inform the stereo kernel that it is no longer available. A new buffer will be created the next time it is required by the stereo kernel.

Parameters

context	The context.

camera	The camera. Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid. InvalidCamera — The input camera is invalid for this camera configuration, or is not associated with the requested image type.

See Also
triclopsRectifyColorImage(), triclopsSetColorImageBuffer()

6.9.6. triclopsUnsetImage16Buffer

Declaration
TriclopsError
triclopsUnsetImage16Buffer(TriclopsContext context,
TriclopsImage16Type imageType,
TriclopsCamera camera)

This releases the user specified internal image buffer for the specified camera and image type. The next time this buffer is required by the system, it will allocate a new one for internal use.

If the user has already called triclopsSetImage16Buffer for a particular camera and image type, the stereo kernel will be using that buffer for internal processing. If the user no longer wants to have the supplied buffer used by the stereo kernel, he/she may use this function to inform the stereo kernel that it is no longer available. A new buffer will be created the next time it is required by the stereo kernel.

Parameters

context	The context.
imageType	The image type.
camera	The camera. Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid. InvalidCamera — The input camera is invalid for this camera configuration, or is not associated with the requested image type.

See Also
triclopsGetImage(), triclopsSetImage16Buffer()

6.9.7. triclopsUnsetImageBuffer

Declaration
TriclopsError
triclopsUnsetImageBuffer(TriclopsContext context,

TriclopsImageType imageType,
TriclopsCamera camera)

This releases the user specified internal image buffer for the specified camera and image type. The next time this buffer is required by the system, it will allocate a new one for internal use.

If the user has already called triclopsSetImageBuffer for a particular camera and image type, the stereo kernel will be using that buffer for internal processing. If the user no longer wants to have the supplied buffer used by the stereo kernel, he/she may use this function to inform the stereo kernel that it is no longer available. A new buffer will be created the next time it is required by the stereo kernel.

Parameters

context	The context.
imageType	The image type.
camera	The camera. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.
InvalidCamera	The input camera is invalid for this camera configuration, or is not associated with the requested image type.

See Also
TriclopsGetImage(), triclopsSetImageBuffer()

6.9.8. triclopsUnsetPackedColorImageBuffer

Declaration
TriclopsError
triclopsUnsetPackedColorImageBuffer(TriclopsContext context,
TriclopsCamera camera)

This releases the user specified internal color image buffer for the specified camera. The next time this buffer is required by the system, it will allocate a new one for internal use.

If the user has already called triclopsSetPackedColorImageBuffer for a particular camera, the stereo kernel will be using that buffer when rectifying a color image. If the user no longer wants to have the supplied buffer used by the stereo kernel, he/she may use this function to inform the stereo kernel that it is no longer available. A new buffer will be created the next time it is required by the stereo kernel.

Parameters

context	The context.
camera	The camera. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.
InvalidCamera	The input camera is invalid for this camera configuration,

		or is not associated with the requested image type.

See Also
triclopsSetPackedColorImageBuffer(), triclopsRectifyPackedColorImage()

6.10. Image I/O Operations

6.10.1. triclopsReadImage16Extra

Declaration
TriclopsError
triclopsReadImage16Extra(char* filename,
TriclopsImage16* triclopsImage16,
TriclopsImageInfo* imageInfo)
Reads an image from a file named "filename" into a TriclopsImage16, filling in as much of the imageInfo structure as possible from the input file's header. Currently, the only input file format supported is PGM.

Reads a 16-bit image from a file named "filename" into a triclopsImage. If the image was created using the triclopsSaveImage16Extra function, it should contain information about the image, depending on the image type. This information is parsed from the header and placed into the imageInfo structure. If the image is a rectified, edge, or disparity image, the appropriate parameter in the TriclopsImageInfo union will be filled in. ** NOTE: Even if the function returns triclopsErrorOk, you still need to check the TriclopsImageInfo "commentFound" boolean field, in case there was no PGRComment associated with the input file.

Parameters

filename	The name of the file from which the image is read.
triclopsImage16	The 16-bit image that gets read.
imageInfo	Relevant information about the image that is read. Returns:

TriclopsErrorOk	The operation succeeded.
TriclopsErrorFileRead	Could not read in the specified file (not found, or corrupted).
TriclopsErrorCorruptPGRComment	The PGR comment in the header was corrupted. The imageInfo is invalid, but the file body ("the image") was read in successfully.

See Also
triclopsReadImageExtra()

6.10.2. triclopsReadImageExtra

Declaration

TriclopsError
triclopsReadImageExtra(char* filename,
TriclopsImage* triclopsImage,
TriclopsImageInfo* imageInfo)

Reads an image from a file named "filename" into a triclopsImage, filling in as much of the imageInfo structure as possible from the input file's header. Currently, the only input file format supported is PGM.

Reads an image from a file named "filename" into a triclopsImage. If the image was created using the triclopsSaveImageExtra function, it should contain information about the image, depending on the image type. This information is parsed from the header and placed into the imageInfo structure. If the image is a rectified, edge, or disparity image, the appropriate parameter in the TriclopsImageInfo union will be filled in. ** NOTE: Even if the function returns triclopsErrorOk, you still need to check the TriclopsImageInfo "commentFound" boolean field, in case there was no PGRComment associated with the input file.

Parameters

filename	The name of the file from which the image is read.
triclopsImage	The image that gets read.
imageInfo	Relevant information about the image that is read. Returns:

TriclopsErrorOk	The operation succeeded.
TriclopsErrorFileRead	Could not read in the specified file (not found, or corrupted).
TriclopsErrorCorruptPGRComment	The PGR comment in the header was corrupted. The imageInfo is invalid, but the file body ("the image") was read in successfully.

See Also
triclopsReadImage16Extra()

6.10.3. triclopsSaveColorImage

Declaration
TriclopsError
triclopsSaveColorImage(TriclopsColorImage* image,
char* filename)

Saves an image to the specified filename. The file format currently supported is PGM format.

This function saves the input image to the requested file. Currently, this function will not detect if the file could not be opened, and will always return successful. Color images are saved in PPM format.

Parameters

image	The TriclopsColorImage to be saved.

filename	The file name in which to save the image. Returns: SystemError — The file could not be opened TriclopsErrorOk — The operation succeeded.

See Also
triclopsSaveImage()

6.10.4. triclopsSaveImage16Extra

Declaration
TriclopsError
triclopsSaveImage16Extra(const TriclopsContext context,
TriclopsImage16Type image16Type,
TriclopsCamera camera,
char* filename)

Saves the specified type of 16-bit image associated with the specified camera to a file named "filename". Currently, the only output file format supported is PGM.

Saves the specified type of 16-bit image associated with the specified camera to a file named "filename". Depending on the image type, certain useful comments are filled in to the header. For all images, the camera type and serial number are filled in.

Parameters

context	The context.
image16Type	The 16-bit image type requested.
camera	The camera that generated the requested image.
filename	The file name in which to save the image. Returns: TriclopsErrorOk — The operation succeeded. SystemError — The file could not be opened InvalidContext — The input context was invalid. InvalidCamera — The camera does not match a camera in this configuration. InvalidRequest — The image type does not match the camera or is not being generated by this system with the current context options. ErrorUnknown — The image was unable to be written properly for an unknown reason.

See Also
triclopsSaveImageExtra()

6.10.5. triclopsSaveImageExtra

Declaration
TriclopsError
triclopsSaveImageExtra(const TriclopsContext context,

TriclopsImageType imageType,
TriclopsCamera camera,
char* filename)

Saves the specified type of image associated with the specified camera to a file named "filename". Currently, the only output file format supported is PGM.

Saves the specified type of image associated with the specified camera to a file named "filename". Depending on the image type, certain useful comments are filled in to the header. For all images, the camera type and serial number are filled in. For each image type, only relevant parameters are filled in. E.g., for the TriImg_RECTIFIED type, the only additional comment is the rectification quality.

Parameters

context	The context.
imageType	The image type requested.
camera	The camera that generated the requested image.
filename	The file name in which to save the image. Returns:

TriclopsErrorOk	The operation succeeded.
SystemError	The file could not be opened
InvalidContext	The input context was invalid.
InvalidCamera	The camera does not match a camera in this configuration.
InvalidRequest	The image type does not match the camera or is not being generated by this system with the current context options.
ErrorUnknown	The image was unable to be written properly for an unknown reason.

6.10.6. triclopsSavePackedColorImage

Declaration
TriclopsError
triclopsSavePackedColorImage(TriclopsPackedColorImage* image,
char* filename)

Allows the user to save a packed color image to the given file.

Parameters

image	A pointer to the buffer containing the image.
filename	The name of the file to be written to. Returns:

SystemError	The file could not be opened
TriclopsErrorOk	Upon the successful completion of the operation.

6.11. Rectification and Camera Geometry

6.11.1. triclopsGetResolution

Declaration
TriclopsError
triclopsGetResolution(const TriclopsContext context,
int* nrows,
int* ncols)
Retrieves the resolution of the resultant images. This includes rectified, disparity and edge images.

This returns the current resolution for output images of the given context.

Parameters

context	The context.
nrows	Number of rows in the output images.
ncols	Number of columns in the output images. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsSetResolution()

6.11.2. triclopsGetSourceResolution

Declaration
TriclopsError
triclopsGetSourceResolution(const TriclopsContext context,
int* nSrcRows,
int* nSrcCols)
Retrieves the resolution of the resultant images. This includes rectified, disparity and edge images.

This returns the current resolution for raw images of the given context.

Parameters

context	The context.
nSrcRows	Number of rows in the raw images.
nSrcCols	Number of columns in the raw images. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsSetSourceResolution()

6.11.3. triclopsSetResolution

Declaration
TriclopsError
triclopsSetResolution(TriclopsContext context,
int nrows,
int ncols)
Sets the resolution of the resultant images. This includes rectified, disparity and edge images.

This function sets the desired resolution of the output images. These images include the rectified, disparity and edge images. This resolution must maintain the 640x480 columns to rows ratio. If the user wishes to have an image of a different aspect ratio, he/she must use Regions Of Interest to control the size of the image that is being processed.

Parameters

context	The context.
nrows	Number of rows in the output images.
ncols	Number of columns in the output images. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.
InvalidSetting	The aspect ratio of the requested image was not 4 columns to 3 rows, or was a negative size.

See Also
triclopsGetResolution(), triclopsGetROIs()

6.11.4. triclopsSetResolutionAndPrepare

Declaration
TriclopsError
triclopsSetResolutionAndPrepare(TriclopsContext context,
int nrows,
int ncols,
int nInputRows,
int nInputCols)
Sets the resolution of the resultant images. This includes rectified, disparity and edge images.

This function sets the desired resolution of the output images and also immediately constructs the rectification tables. For large images, the construction of the rectification can take a while. This function allows you to control when the construction takes place, otherwise it will occur during the first call to triclopsPreprocess(). The resolution of the input images must be specified at this time, as this is necessary for the construction of the tables. The output images include the rectified, disparity and edge images. This requested resolution must maintain the 640x480 columns to rows ratio. If the user wishes to have an image of a different aspect ratio, he/she must use Regions Of Interest to control the size of the image that is being processed. For feature based stereo application where

rectification of the entire image is not needed, one should call triclopsSetResolution() and triclopsSetSourceResolution() only (these are much simpler functions), and then simply proceeds to call triclopsRectifyPixel() and triclopsUnrectifyPixel() for the small set of feature pixels needed.

Parameters

context	The context.
nrows	Number of rows in the output images.
ncols	Number of columns in the output images.
nInputRows	Number of rows in the input images.
nInputCols	Number of columns in the input images. Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid. InvalidSetting — The aspect ratio of the requested image was not 4 columns to 3 rows, or was of negative size.

See Also
triclopsGetResolution(), triclopsSetResolution(), triclopsSetSourceResolution(), triclopsSetRectify()

6.11.5. triclopsSetSourceResolution

Declaration
TriclopsError
triclopsSetSourceResolution(TriclopsContext context,
int nSrcRows,
int nSrcCols)
Sets the resolution of the raw images that the library will be processing later.

This function sets the expected resolution of the raw images. This function is provided primarily to support feature based stereo application where one is expected to make direct calls to triclopsRectifyPixel() and triclopsUnrectifyPixel() on a point by point basis. For regular stereo application where an entire image will be rectified every time, the application should use triclopsSetResolutionAndPrepare() which in addition to setting up both the source and rectification resolution, it also creates the rectification table to speed up the full image rectification calls.

Parameters

context	The context.
nSrcRows	Number of rows in the raw images.
nSrcCols	Number of columns in the raw images. Returns: TriclopsErrorOk — The operation succeeded. InvalidContext — The input context was invalid. InvalidSetting — The aspect ratio of the requested image was not 4 columns to 3 rows, or was a negative size.

See Also
triclopsGetSourceResolution(), triclopsRectifyPixel(), triclopsUnrectifyPixel(), triclopsSetResolutionAndPrepare()

6.12. Rectification

6.12.1. triclopsGetLowpass

Declaration
TriclopsError
triclopsGetLowpass(const TriclopsContext context,
TriclopsBool* on)
Retrieves the state of the low-pass filtering feature.

Parameters

context	The context.
on	A pointer to a Boolean variable that will store the current setting. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsSetLowpass()

6.12.2. triclopsGetRectify

Declaration
TriclopsError
triclopsGetRectify(const TriclopsContext context,
TriclopsBool* on)
Retrieves the state of the rectification feature.

Parameters

context	The context.
on	A pointer to a Boolean that will store the current setting. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsSetRectify(), triclopsSetRectImgQuality(), triclopsGetRectImgQuality()

6.12.3. triclopsGetRectImgQuality

Declaration
TriclopsError
triclopsGetRectImgQuality(TriclopsContext context,
TriclopsRectImgQuality* quality)

Gets the quality of the algorithm currently in use for image rectification.

Parameters

context	The current TriclopsContext.
quality	The quality of the rectification algorithm currently in use. Returns:

TriclopsErrorOk	The operation succeeded.
TriclopsErrorInvalidContext	If the context is invalid.

See Also
triclopsSetRectImgQuality()

6.12.4. triclopsPreprocess

Declaration
TriclopsError
triclopsPreprocess(TriclopsContext context,
TriclopsInput* input)
Does image unpacking, smoothing, rectification and edge detection, as specified by parameters.

This function does all necessary preprocessing on the input data. It unpacks the data, which strips individual channels from 32-bit packed data and puts them into 3 TriImg_RAW channels. It applies a low-pass filter on these channels if requested, and corrects for lens distortion and camera misalignment, saving these images into TriImg_RECTIFIED. Finally it performs 2nd derivative Gaussian edge processing on the rectified images and saves them into TriImg_EDGE internal images.

Parameters

context	The context.
input	The image to be processed. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.
NotImplemented	TriclopsInput has rowinc < ncols*pixelsize

Remarks
This function has been renamed as "triclopsRectify()", which is a more descriptive name. "triclopsPreprocess()" will be deprecated in the future.

See Also
triclopsStereo(), triclopsSetResolution(), triclopsSetEdgeCorrelation(), triclopsSetRectify(), triclopsSetLowpass()

6.12.5. triclopsRectify

Declaration

TriclopsError
triclopsRectify(TriclopsContext context,
TriclopsInput* input)
Does image unpacking, smoothing, rectification and edge detection, as specified by parameters.

This function does all necessary preprocessing on the input data. It unpacks the data, which strips individual channels from 32-bit packed data and puts them into 3 TriImg_RAW channels. It applies a low-pass filter on these channels if requested, and corrects for lens distortion and camera misalignment, saving these images into TriImg_RECTIFIED. Finally it performs 2nd derivative Gaussian edge processing on the rectified images and saves them into TriImg_EDGE internal images.

Parameters

context	The context.
input	The image to be processed. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.
NotImplemented	TriclopsInput has rowinc < ncols*pixelsize

See Also
triclopsStereo(), triclopsSetResolution(), triclopsSetEdgeCorrelation(), triclopsSetRectify(), triclopsSetLowpass()

6.12.6. triclopsRectifyColorImage

Declaration
TriclopsError
triclopsRectifyColorImage(TriclopsContext context,
TriclopsCamera nCamera,
TriclopsInput* input,
TriclopsColorImage* output)
Rectifies a TriclopsInput structure into a TriclopsColorImage. This function is used for TriclopsInput's that have been obtained from a Color stereo camera product. If the system is a Color Triclops, nCamera should be set to TriCam_COLOR.

Parameters

context	The context.
nCamera	A TriclopsCamera enumerated value indicating which camera the input image came from.
input	The raw color image encoded into a TriclopsInput structure.
output	The resultant rectified color image. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.
InvalidRequest	The input raw image had a size of 0.
InvalidCamera	There is a corruption in the color camera calibration data

		in the calibration file.
	InvalidSettings	The TriclopsInput structure did not have a recognizable data type.

6.12.7. triclopsRectifyPackedColorImage

Declaration
TriclopsError
triclopsRectifyPackedColorImage(TriclopsContext context,
TriclopsCamera nCamera,
TriclopsInput* input,
TriclopsPackedColorImage* output)
This function rectifies a packed color image. This function will only rectify a packed TriclopsInput (a TriclopsInput of type TriInp_RGB_32BIT_PACKED). It is useful for creating rectified color images for display, since bitmap displays commonly require the data format to be packed.

Parameters

context	The TriclopsContext to use to rectify the color image.
nCamera	The camera from which this TriclopsInput originated. If the system is a Color Triclops, nCamera should be set to TriCam_COLOR.
input	The color image to be rectified.
output	The rectified color image. Returns: TriclopsErrorOk — Upon the successful completion of the operation.

6.12.8. triclopsRectifyPixel

Declaration
TriclopsError
triclopsRectifyPixel(const TriclopsContext context,
TriclopsCamera camera,
float rowIn,
float colIn,
float* rowOut,
float* colOut)
Converts a pixel coordinate location from the unrectified image coordinates to rectified image coordinates. The source image dimension must have been previously set either via a call to triclopsSetSourceResolution() or triclopsSetResolutionAndPrepare().

Parameters

context	The context.
camera	The camera for which the pixel should be rectified.
rowIn	The location of the pixel to rectify.
colIn	The location of the pixel to rectify.
rowOut	The location of the rectified pixel.

colOut	The location of the rectified pixel. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.
InvalidCamera	The camera argument was invalid.
TriclopsErrorInvalidSetting	The raw image dimension has not been set

See Also
triclopsPreprocess(), triclopsRectifyColorImage(), triclopsSetSourceResolution()

6.12.9. triclopsSetLowpass

Declaration
TriclopsError
triclopsSetLowpass(TriclopsContext context,
TriclopsBool on)
Turns low-pass filtering before rectification on or off.

Parameters

context	The context.
on	A Boolean value indicating whether it should be turned on or off. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsGetLowpass()

6.12.10. triclopsSetRectify

Declaration
TriclopsError
triclopsSetRectify(TriclopsContext context,
TriclopsBool on)
Turns rectification on or off.

Parameters

context	The context.
on	A Boolean indicating whether rectification should be turned on or off. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.

See Also
triclopsGetRectify(), triclopsSetRectImgQuality(), triclopsGetRectImgQuality()

6.12.11. triclopsSetRectImgQuality

Declaration

TriclopsError
triclopsSetRectImgQuality(TriclopsContext context,
TriclopsRectImgQuality quality)

Sets the quality of the algorithm to use during image rectification. Higher-quality algorithms will generally require more processing time, but give more accurate results. Currently available are TriRectQlty_STANDARD, our classic rectification algorithm, and TriRectQlty_ENHANCED, which uses a more elaborate kernel in the rectification process.

Parameters

context	The current TriclopsContext.
quality	The desired quality of the rectification algorithm to use. Returns:

TriclopsErrorOk	The operation succeeded.
TriclopsErrorInvalidContext	If the context is invalid.

See Also
triclopsGetRectImgQuality()

6.12.12. triclopsUnrectifyPixel

Declaration
TriclopsError
triclopsUnrectifyPixel(const TriclopsContext context,
TriclopsCamera camera,
float rowIn,
float colIn,
float* rowOut,
float* colOut)

Converts a pixel coordinate location from the rectified image coordinates to unrectified image coordinates.

Parameters

context	The context.
camera	The camera for which the pixel should be unrectified.
rowIn	The location of the pixel to unrectify.
colIn	The location of the pixel to unrectify.
rowOut	The location of the unrectified pixel.
colOut	The location of the unrectified pixel. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context was invalid.
InvalidCamera	The camera argument was invalid.
TriclopsErrorInvalidSetting	The raw image dimension has not been set
InvalidRequest	The requested rectified location cannot be unrectified into a location that is within the proper raw image bounds

Remarks
This version will accommodate input rectified pixel locations that are outside of the normal rectified image bounds, as long as the corresponding unrectified location is within its own image bounds. The source image dimension must have been previously set either via a call to triclopsSetSourceResolution() or triclopsSetResolutionAndPrepare().

See Also
triclopsRectifyPixel(), triclopsSetSourceResolution()

6.13. Triclops Context Manipulation

6.13.1. triclopsCopyContext

Declaration
TriclopsError
triclopsCopyContext(const TriclopsContext contextIn,
TriclopsContext* contextOut)
Copies the context parameters and images. Care must be taken when using triclopsCopyContext() as contexts will share image buffers.

This function creates a copy of the input context. This allows the user to have two contexts with the same options and also to share the same image buffers. Care must be taken with this function. The ensuing contexts will share image buffers. This allows them to share some of the previous processing, however, any changes in resolution or calls to set image buffers will create new buffers to be used. These buffers will no longer be shared and thus programs may not operate as expected at this time. The recommendation is to use this operation when one wants to run stereo with different stereo options. After the preprocessing step, the context can be copied and different stereo options set before the call to triclopsStereo().

Note: This only copies the options, not the images

Parameters

contextIn	A context that has already been constructed.
contextOut	A copy of the input context. Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The input context is invalid.

See Also
triclopsGetDefaultContext(), triclopsDestroyContext()

6.13.2. triclopsDestroyContext

Declaration
TriclopsError
triclopsDestroyContext(TriclopsContext context)
Destroys the given context.

Input:

Context	The context to be destroyed.

Returns:

TriclopsErrorOk	The operation succeeded.
InvalidContext	The context was invalid.

This function destroys a context and frees all associated memory.

6.13.3. triclopsGetDefaultContextFromFile

Declaration
TriclopsError
triclopsGetDefaultContextFromFile(TriclopsContext* defaultContext,
char* filename)
Setup the initial context with data obtained from a file.

This function reads in the default option list and the camera calibration data from a file. If the specified file is not found, contains invalid fields, or is for the wrong product, the value of CorruptConfigFile will be returned.

Parameters

defaultContext	The default context. Returns: TriclopsErrorOk: The operation succeeded. CorruptConfigFile: The specified file was corrupt, or was for the wrong Triclops version.
filename	The configuration file name.

See Also
triclopsWriteDefaultContextToFile(), triclopsWriteCurrentContextToFile()

6.13.4. triclopsWriteCurrentContextToFile

Declaration
TriclopsError
triclopsWriteCurrentContextToFile(TriclopsContext context,
char* filename)
This writes the calibration and parameter file from the TriclopsContext to a file. It is the "current" context, so it uses the parameters that are currently active in the TriclopsContext. Any parameter changes that have been made through the API will be reflected in this calibration file.

This function writes the current context parameters and calibration data to a file.

Parameters

context	The TriclopsContext
filename	The name of the file to be written

See Also

TriclopsGetDefaultContextFromFile(), triclopsWriteDefaultContextToFile()

6.13.5. triclopsWriteDefaultContextToFile

Declaration
TriclopsError
triclopsWriteDefaultContextToFile(TriclopsContext context,
char* filename)
This writes the default calibration file from the TriclopsContext to a file.

This function writes the default context parameters and calibration data to a file. It does not write the current configuration, rather the original configuration that would obtained when getting the default from either a file or a device.

Parameters

context	The TriclopsContext
filename	The name of the file to be written

See Also
TriclopsGetDefaultContextFromFile(), triclopsWriteCurrentContextToFile()

6.14. Ungrouped Objects

6.14.1. TRICLOPS_VERSION

Declaration
#define TRICLOPS_VERSION 3310
Remarks
The version of the library.

7. Contacting Point Grey Research

For any questions, concerns or comments please contact us via the following methods:

Email:	For all general questions about Point Grey Research please contact us at info@ptgrey.com. For technical support (existing customers only) contact us at http://www.ptgrey.com/support/contact/.	
Knowledge Base:	Find answers to commonly asked questions in our knowledge base at http://www.ptgrey.com/support/kb/.	
Downloads:	Users can download the latest manuals and software from http://www.ptgrey.com/support/downloads/	
Main Office:	**Mailing Address:** Point Grey Research, Inc. 12051 Riverside Way Richmond, BC, Canada V6W 1K7	**Tel:** +1 (604) 242-9937 **Toll-free** (North America only)**:** +1 (866) 765-0827 **Fax:** +1 (604) 242-9938 sales@ptgrey.com
Distributors		
USA	**Tel:** +1 (866) 765-0827 na-sales@ptgrey.com	
Europe Israel	**Mailing Address:** Point Grey Research GmbH Schwieberdinger Strasse 60 71636 Ludwigsburg Germany	**Tel:** +49 7141 488817-0 **Fax:** +49 7141 488817-99 eu-sales@ptgrey.com
Japan	ViewPLUS Inc. (http://www.viewplus.co.jp/)	
Korea	Cylod Co. Ltd. (http://www.cylod.com)	
China	LUSTER LightVision Tech. Co., Ltd (www.lusterlighttech.com)	
Singapore Malaysia Thailand	Voltrium Systems Pte Ltd. (www.voltrium.com.sg)	
Taiwan	Apo Star Co., Ltd. (www.apostar.com.tw)	

INDEX

API. See Application Programming Interface
Application Programming Interface
  API function reference, 22
  Programming with the Triclops API, 20
blue. *See* TriclopsInputRGB::blue, *See* TriclopsColorImage::blue
Calculating distances, 14
col. *See* TriclopsROI::col
Compiling a Triclops Program, 12
Coordinate systems, 15
Correlation mask, 17
data. *See* TriclopsPackedColorImage::data, *See* TriclopsInputRGB32BitPacked::data, *See* TriclopsImage16::data, *See* TriclopsImage::data
Disparity, 14
Disparity range, 17, 20
DisparityImageInfo, 27
Edge detection, 16
EdgeImageInfo, 27
Establishing correspondence, 14
Examples, 21
green. *See* TriclopsInputRGB::green, *See* TriclopsColorImage::green
Image
  size/resolution, 20
  Viewing, 11
inputType. *See* TriclopsInput::inputType
Installation
  Software, 10
License Agreement, 2
Low-pass filtering, 16
Mask Size, 20
Matching points, 13
Multiple cameras, 15
ncols. *See* TriclopsROI::ncols, *See* TriclopsPackedColorImage::ncols, *See* TriclopsInput::ncols, *See* TriclopsImage3d::ncols, *See* TriclopsImage16::ncols, *See* TriclopsImage::ncols, *See* TriclopsColorImage::ncols
nrows. *See* TriclopsROI::nrows, *See* TriclopsPackedColorImage::nrows, *See* TriclopsInput::nrows, *See* TriclopsImage3d::nrows, *See* TriclopsImage16::nrows, *See* TriclopsImage::nrows, *See* TriclopsColorImage::nrows
Online Resources, 10
point. *See* TriclopsPoint3d::point
points. *See* TriclopsImage3d::points
Preprocessing, 16, 20
Range measurement, 9, 13
Rectification, 16
RectifiedImageInfo, 28
red. *See* TriclopsInputRGB::red, *See* TriclopsColorImage::red
Regions of interest, 20
Requirements. See System requirements
ROI. See Regions of interest
row. *See* TriclopsROI::row
rowinc. *See* TriclopsPackedColorImage::rowinc, *See* TriclopsInput::rowinc, *See* TriclopsImage3d::rowinc, *See* TriclopsImage16::rowinc, *See* TriclopsImage::rowinc, *See* TriclopsColorImage::rowinc
Running the Demo Program, 12
sec. *See* TriclopsTimestamp::sec
Source Code. See Examples
Stereo context, 21
Stereo processing, 17
Stereo Vision Parameters
  Disparity range, 20
  Image size/resolution, 20
  Mask Size, 20
  Preprocessing, 20
  Regions of interest, 20
  Sub-pixel interpolation, 20
  Validation, 20
Stereo vision technology, 9
Sub-pixel interpolation, 18, 20

Sum of Absolute Differences, 14, 17
System requirements, 10
Texture validation, 18
timeStamp. *See*
  TriclopsInput::timeStamp
Triangulation, 9
TriCam_COLOR. *See*
  TriclopsCamera::TriCam_COLOR
TriCam_L_COLOR. *See*
  TriclopsCamera::TriCam_L_COLOR
TriCam_L_LEFT. *See*
  TriclopsCamera::TriCam_L_LEFT
TriCam_L_RIGHT. *See*
  TriclopsCamera::TriCam_L_RIGHT
TriCam_L_TOP. *See*
  TriclopsCamera::TriCam_L_TOP
TriCam_LEFT. *See*
  TriclopsCamera::TriCam_LEFT
TriCam_REFERENCE. *See*
  TriclopsCamera::TriCam_REFEREN
  CE
TriCam_RIGHT. *See*
  TriclopsCamera::TriCam_RIGHT
TriCam_TOP. *See*
  TriclopsCamera::TriCam_TOP
TriCfg_2CAM_HORIZONTAL. *See*
  TriclopsCameraConfiguration::TriCfg
  _2CAM_HORIZONTAL
TriCfg_2CAM_HORIZONTAL_NARR
  OW. *See*
  TriclopsCameraConfiguration::TriCfg
  _2CAM_HORIZONTAL_NARROW
TriCfg_2CAM_HORIZONTAL_WIDE.
  See
  TriclopsCameraConfiguration::TriCfg
  _2CAM_HORIZONTAL_WIDE
TriCfg_2CAM_VERTICAL. *See*
  TriclopsCameraConfiguration::TriCfg
  _2CAM_VERTICAL
TriCfg_L. *See*
  TriclopsCameraConfiguration::TriCfg
  _L
TriCfg_MAX_VALUE. *See*
  TriclopsCameraConfiguration::TriCfg
  _MAX_VALUE
TRICLOPS_VERSION, 99
TriclopsBool, 28
triclopsBuildPackedTriclopsInput, 47
triclopsBuildRGBTriclopsInput, 47
TriclopsCamera, 22
  TriCam_COLOR, 22
  TriCam_L_COLOR, 22
  TriCam_L_LEFT, 22
  TriCam_L_RIGHT, 22
  TriCam_L_TOP, 22
  TriCam_LEFT, 22
  TriCam_REFERENCE, 22
  TriCam_RIGHT, 22
  TriCam_TOP, 22
TriclopsCameraConfiguration, 23
  TriCfg_2CAM_HORIZONTAL, 23
  TriCfg_2CAM_HORIZONTAL_NAR
    ROW, 23
  TriCfg_2CAM_HORIZONTAL_WID
    E, 23
  TriCfg_2CAM_VERTICAL, 23
  TriCfg_L, 23
  TriCfg_MAX_VALUE, 23
TriclopsColorImage, 28
  blue, 28
  green, 28
  ncols, 28
  nrows, 28
  red, 28
  rowinc, 28
TriclopsContext, 29
triclopsCopyContext, 97
triclopsCreateImage3d, 64
triclopsDestroyContext, 97
triclopsDestroyImage3d, 65
TriclopsError, 23
  TriclopsErrorBadOptions, 23
  TriclopsErrorCorruptConfigFile, 23
  TriclopsErrorCorruptPGRComment,
    23
  TriclopsErrorCorruptTransformFile,
    23
  TriclopsErrorDeprecated, 23
  TriclopsErrorFileRead, 23
  TriclopsErrorInvalidCamera, 23
  TriclopsErrorInvalidContext, 23
  TriclopsErrorInvalidParameter, 23

TriclopsErrorInvalidRequest, 23
TriclopsErrorInvalidROI, 23
TriclopsErrorInvalidSetting, 23
TriclopsErrorNoConfigFile, 23
TriclopsErrorNonMMXCpu, 23
TriclopsErrorNotImplemented, 23
TriclopsErrorOk, 23
TriclopsErrorSurfaceValidationOverflow, 23
TriclopsErrorSystemError, 23
TriclopsErrorUnknown, 23
TriclopsErrorBadOptions. *See* TriclopsError::TriclopsErrorBadOptions
TriclopsErrorCorruptConfigFile. *See* TriclopsError::TriclopsErrorCorruptConfigFile
TriclopsErrorCorruptPGRComment. *See* TriclopsError::TriclopsErrorCorruptPGRComment
TriclopsErrorCorruptTransformFile. *See* TriclopsError::TriclopsErrorCorruptTransformFile
TriclopsErrorDeprecated. *See* TriclopsError::TriclopsErrorDeprecated
TriclopsErrorFileRead. *See* TriclopsError::TriclopsErrorFileRead
TriclopsErrorInvalidCamera. *See* TriclopsError::TriclopsErrorInvalidCamera
TriclopsErrorInvalidContext. *See* TriclopsError::TriclopsErrorInvalidContext
TriclopsErrorInvalidParameter. *See* TriclopsError::TriclopsErrorInvalidParameter
TriclopsErrorInvalidRequest. *See* TriclopsError::TriclopsErrorInvalidRequest
TriclopsErrorInvalidROI. *See* TriclopsError::TriclopsErrorInvalidROI
TriclopsErrorInvalidSetting. *See* TriclopsError::TriclopsErrorInvalidSetting
TriclopsErrorNoConfigFile. *See* TriclopsError::TriclopsErrorNoConfigFile
TriclopsErrorNonMMXCpu. *See* TriclopsError::TriclopsErrorNonMMXCpu
TriclopsErrorNotImplemented. *See* TriclopsError::TriclopsErrorNotImplemented
TriclopsErrorOk. *See* TriclopsError::TriclopsErrorOk
TriclopsErrorSurfaceValidationOverflow. *See* TriclopsError::TriclopsErrorSurfaceValidationOverflow
TriclopsErrorSystemError. *See* TriclopsError::TriclopsErrorSystemError
triclopsErrorToString, 35
TriclopsErrorUnknown. *See* TriclopsError::TriclopsErrorUnknown
triclopsExtractImage3d, 65
triclopsExtractWorldImage3d, 66
triclopsGetBackForthValidation, 35
triclopsGetBackForthValidationMapping, 36
triclopsGetBaseline, 61
triclopsGetCameraConfiguration, 62
triclopsGetDefaultContextFromFile, 98
triclopsGetDeviceConfiguration, 62
triclopsGetDisparity, 51
triclopsGetDisparityMapping, 51
triclopsGetDisparityMappingOn, 51
triclopsGetDisparityOffset, 52
triclopsGetDoStereo, 52
triclopsGetEdgeCorrelation, 53
triclopsGetEdgeMask, 53
triclopsGetFocalLength, 63
triclopsGetImage, 48
triclopsGetImage16, 49
triclopsGetImageCenter, 63
triclopsGetLowpass, 91
triclopsGetMaxThreadCount, 53
triclopsGetRectify, 91
triclopsGetRectImgQuality, 91
triclopsGetResolution, 88

triclopsGetROIs, 54
triclopsGetSerialNumber, 64
triclopsGetSourceResolution, 88
triclopsGetStereoMask, 54
triclopsGetStereoQuality, 55
triclopsGetStrictSubpixelValidation, 36
triclopsGetSubpixelInterpolation, 55
triclopsGetSubpixelValidationMapping, 36
triclopsGetSurfaceValidation, 37
triclopsGetSurfaceValidationDifference, 37
triclopsGetSurfaceValidationMapping, 38
triclopsGetSurfaceValidationSize, 38
triclopsGetTextureValidation, 38
triclopsGetTextureValidationMapping, 39
triclopsGetTextureValidationThreshold, 39
triclopsGetTransformFromFile, 66
triclopsGetTriclopsToWorldTransform, 67
triclopsGetUniquenessValidation, 39
triclopsGetUniquenessValidationMapping, 40
triclopsGetUniquenessValidationThreshold, 40
TriclopsImage, 29
  data, 29
  ncols, 29
  nrows, 29
  rowinc, 29
TriclopsImage16, 29
  data, 29
  ncols, 29
  nrows, 29
  rowinc, 29
TriclopsImage16Type, 25
  TriImg16_DISPARITY, 25
TriclopsImage3d, 30
  ncols, 30
  nrows, 30
  points, 30
  rowinc, 30
TriclopsImageInfo, 30
TriclopsImageType, 25
  TriImg_DISPARITY, 25
  TriImg_EDGE, 25
  TriImg_RAW, 25
  TriImg_RECTIFIED, 25
TriclopsInput, 31
  inputType, 31
  ncols, 31
  nrows, 31
  rowinc, 31
  timeStamp, 31
  u, 31
TriclopsInputRGB, 31
  blue, 31
  green, 31
  red, 31
TriclopsInputRGB32BitPacked, 32
  data, 32
TriclopsInputType, 26
  TriInp_NONE, 26
  TriInp_RGB, 26
  TriInp_RGB_32BIT_PACKED, 26
TriclopsPackedColorImage, 32
  data, 32
  ncols, 32
  nrows, 32
  rowinc, 32
TriclopsPackedColorPixel, 33
  value, 33
TriclopsPoint3d, 33
  point, 33
triclopsPreprocess, 92
triclopsRCD16ToWorldXYZ, 67
triclopsRCD16ToXYZ, 68
triclopsRCD8ToWorldXYZ, 69
triclopsRCD8ToXYZ, 70
triclopsRCDFloatToWorldXYZ, 71
triclopsRCDFloatToXYZ, 72
triclopsRCDMappedToWorldXYZ, 72
triclopsRCDMappedToXYZ, 73
triclopsRCDToWorldXYZ, 74
triclopsRCDToXYZ, 75
triclopsReadImage16Extra, 84
triclopsReadImageExtra, 84
triclopsRectify, 92
triclopsRectifyColorImage, 93

triclopsRectifyPackedColorImage, 94
triclopsRectifyPixel, 94
TriclopsRectImgQuality, 33
  TriRectQlty_ENHANCED_1, 33
  TriRectQlty_ENHANCED_2, 33
  TriRectQlty_FAST, 33
  TriRectQlty_STANDARD, 33
TriclopsROI, 33
  col, 33
  ncols, 33
  nrows, 33
  row, 33
triclopsSaveColorImage, 85
triclopsSaveImage, 49
triclopsSaveImage16, 50
triclopsSaveImage16Extra, 86
triclopsSaveImageExtra, 86
triclopsSavePackedColorImage, 87
triclopsSetAnyStereoMask, 55
triclopsSetBackForthValidation, 41
triclopsSetBackForthValidationMapping, 41
triclopsSetCameraConfiguration, 64
triclopsSetColorImageBuffer, 79
triclopsSetDisparity, 56
triclopsSetDisparityMapping, 56
triclopsSetDisparityMappingOn, 57
triclopsSetDoStereo, 57
triclopsSetEdgeCorrelation, 58
triclopsSetEdgeMask, 58
triclopsSetImage16Buffer, 80
triclopsSetImageBuffer, 80
triclopsSetLowpass, 95
triclopsSetMaxThreadCount, 58
triclopsSetNumberOfROIs, 59
triclopsSetPackedColorImageBuffer, 81
triclopsSetRectify, 95
triclopsSetRectImgQuality, 95
triclopsSetResolution, 89
triclopsSetResolutionAndPrepare, 89
triclopsSetSourceResolution, 90
triclopsSetStereoMask, 59
triclopsSetStereoQuality, 60
triclopsSetStrictSubpixelValidation, 42
triclopsSetSubpixelInterpolation, 60
triclopsSetSubpixelValidationMapping, 42
triclopsSetSurfaceValidation, 42
triclopsSetSurfaceValidationDifference, 43
triclopsSetSurfaceValidationMapping, 43
triclopsSetSurfaceValidationSize, 44
triclopsSetTextureValidation, 44
triclopsSetTextureValidationMapping, 44
triclopsSetTextureValidationThreshold, 45
triclopsSetTriclopsToWorldTransform, 76
triclopsSetUniquenessValidation, 45
triclopsSetUniquenessValidationMapping, 46
triclopsSetUniquenessValidationThreshold, 46
triclopsStereo, 61
TriclopsStereoQuality, 34
  TriStereoQlty_ENHANCED, 34
  TriStereoQlty_STANDARD, 34
TriclopsTimestamp, 34
  sec, 34
  u_sec, 34
TriclopsTransform, 35
triclopsUnrectifyPixel, 96
triclopsUnsetColorImageBuffer, 81
triclopsUnsetImage16Buffer, 82
triclopsUnsetImageBuffer, 82
triclopsUnsetPackedColorImageBuffer, 83
triclopsVersion, 50
triclopsWorldXYZToRCD, 77
triclopsWriteCurrentContextToFile, 98
triclopsWriteDefaultContextToFile, 99
triclopsWriteTransformToFile, 78
triclopsXYZToRCD, 78
TriImg_DISPARITY. *See* TriclopsImageType::TriImg_DISPARITY
TriImg_EDGE. *See* TriclopsImageType::TriImg_EDGE

TriImg_RAW. *See* TriclopsImageType::TriImg_RAW
TriImg_RECTIFIED. *See* TriclopsImageType::TriImg_RECTIFIED
TriImg16_DISPARITY. *See* TriclopsImage16Type::TriImg16_DISPARITY
TriInp_NONE. *See* TriclopsInputType::TriInp_NONE
TriInp_RGB. *See* TriclopsInputType::TriInp_RGB
TriInp_RGB_32BIT_PACKED. *See* TriclopsInputType::TriInp_RGB_32BIT_PACKED
TriRectQlty_ENHANCED_1. *See* TriclopsRectImgQuality::TriRectQlty_ENHANCED_1
TriRectQlty_ENHANCED_2. *See* TriclopsRectImgQuality::TriRectQlty_ENHANCED_2
TriRectQlty_FAST. *See* TriclopsRectImgQuality::TriRectQlty_FAST
TriRectQlty_STANDARD. *See* TriclopsRectImgQuality::TriRectQlty_STANDARD
TriStereoQlty_ENHANCED. *See* TriclopsStereoQuality::TriStereoQlty_ENHANCED
TriStereoQlty_STANDARD. *See* TriclopsStereoQuality::TriStereoQlty_STANDARD
Troubleshooting
  Contacting Point Grey Research, 100
u. *See* TriclopsInput::u
u_sec. *See* TriclopsTimestamp::u_sec
Uniqueness validation, 18
Validation, 17, 20
  Texture validation, 18
  Uniqueness validation, 18
value. *See* TriclopsPackedColorPixel::value