

libvisiongl

1.0

Generated by Doxygen 1.7.6.1

Fri Jul 5 2013 07:45:15



# Contents

<b>1</b>	<b>Directory Hierarchy</b>	<b>1</b>
1.1	Directories . . . . .	1
<b>2</b>	<b>File Index</b>	<b>3</b>
2.1	File List . . . . .	3
<b>3</b>	<b>Directory Documentation</b>	<b>5</b>
3.1	src/ Directory Reference . . . . .	5
<b>4</b>	<b>File Documentation</b>	<b>7</b>
4.1	src/glsl2cpp_BG.h File Reference . . . . .	7
4.1.1	Function Documentation . . . . .	7
4.1.1.1	vglDetectFGSimpleBGModel . . . . .	7
4.1.1.2	vglTrainSimpleBGModel . . . . .	7
4.1.1.3	vglUpdatePartialSimpleBGModel . . . . .	8
4.2	src/glsl2cpp_shaders.h File Reference . . . . .	8
4.2.1	Function Documentation . . . . .	10
4.2.1.1	shader_15_1 . . . . .	10
4.2.1.2	vgl1to3Channels . . . . .	10
4.2.1.3	vglAbsDiff . . . . .	10
4.2.1.4	vglAnd . . . . .	10
4.2.1.5	vglBaricenterInit . . . . .	10
4.2.1.6	vglBlurSq3 . . . . .	10
4.2.1.7	vglClear2 . . . . .	10
4.2.1.8	vglContrast . . . . .	10
4.2.1.9	vglCoordToColor . . . . .	11

4.2.1.10	vglCopy . . . . .	11
4.2.1.11	vglCrossingNumber . . . . .	11
4.2.1.12	vglDeleteSkeletonCorners . . . . .	11
4.2.1.13	vglDeleteSkeletonWarts . . . . .	12
4.2.1.14	vglDeleteSkeletonWarts2 . . . . .	12
4.2.1.15	vglDiff . . . . .	12
4.2.1.16	vglDilateCross3 . . . . .	13
4.2.1.17	vglDilateSq3 . . . . .	13
4.2.1.18	vglErodeCross3 . . . . .	13
4.2.1.19	vglErodeHL3 . . . . .	13
4.2.1.20	vglErodeHL5 . . . . .	13
4.2.1.21	vglErodeHL7 . . . . .	13
4.2.1.22	vglErodeSq3 . . . . .	13
4.2.1.23	vglErodeSq3off . . . . .	13
4.2.1.24	vglErodeSq5 . . . . .	14
4.2.1.25	vglErodeSq5off . . . . .	14
4.2.1.26	vglErodeSq7 . . . . .	14
4.2.1.27	vglErodeSqSide . . . . .	14
4.2.1.28	vglErodeVL3 . . . . .	14
4.2.1.29	vglErodeVL5 . . . . .	14
4.2.1.30	vglErodeVL7 . . . . .	14
4.2.1.31	vglFeaturePoints . . . . .	14
4.2.1.32	vglGaussianBlurSq3 . . . . .	15
4.2.1.33	vglGray . . . . .	15
4.2.1.34	vglHorizontalFlip . . . . .	15
4.2.1.35	vglInOut . . . . .	15
4.2.1.36	vglJulia . . . . .	15
4.2.1.37	vglLaplaceSq3 . . . . .	15
4.2.1.38	vglMandel . . . . .	16
4.2.1.39	vglMipmap . . . . .	16
4.2.1.40	vglMulScalar . . . . .	16
4.2.1.41	vglMultiInput . . . . .	16
4.2.1.42	vglMultiOutput . . . . .	16
4.2.1.43	vglNoise . . . . .	16

4.2.1.44	vglNot . . . . .	16
4.2.1.45	vglOr . . . . .	16
4.2.1.46	vglRescale . . . . .	16
4.2.1.47	vglRgbToBgr . . . . .	17
4.2.1.48	vglRgbToHsl . . . . .	17
4.2.1.49	vglRgbToHsv . . . . .	17
4.2.1.50	vglRgbToXyz . . . . .	17
4.2.1.51	vglRobertsGradient . . . . .	17
4.2.1.52	vglSelfSum22 . . . . .	17
4.2.1.53	vglSelfSum3v . . . . .	17
4.2.1.54	vglSelfSum4h . . . . .	17
4.2.1.55	vglSelfSum5h . . . . .	17
4.2.1.56	vglSelfSum5v . . . . .	18
4.2.1.57	vglSharpenSq3 . . . . .	18
4.2.1.58	vglSobelGradient . . . . .	18
4.2.1.59	vglSobelXSq3 . . . . .	18
4.2.1.60	vglSobelYSq3 . . . . .	18
4.2.1.61	vglSum . . . . .	18
4.2.1.62	vglSumWeighted . . . . .	18
4.2.1.63	vglSwapRGB . . . . .	18
4.2.1.64	vglTeste . . . . .	19
4.2.1.65	vglTestInOut . . . . .	19
4.2.1.66	vglTestInOut2 . . . . .	19
4.2.1.67	vglTestMultiInput . . . . .	19
4.2.1.68	vglTestMultiOutput . . . . .	19
4.2.1.69	vglThinBernardAux . . . . .	19
4.2.1.70	vglThinChinAux . . . . .	19
4.2.1.71	vglThresh . . . . .	20
4.2.1.72	vglThreshLevelSet . . . . .	20
4.2.1.73	vglVerticalFlip . . . . .	20
4.2.1.74	vglWhiteRohrerEdge . . . . .	20
4.2.1.75	vglXGY . . . . .	20
4.2.1.76	vglZoom . . . . .	20
4.3	src/gsl2cpp_Stereo.h File Reference . . . . .	21

4.3.1	Function Documentation . . . . .	21
4.3.1.1	vglAbsDiffDisparity . . . . .	21
4.3.1.2	vglAbsDiffDisparityMipmap . . . . .	21
4.3.1.3	vglFindDisparity . . . . .	22
4.3.1.4	vglFindDisparityDiff . . . . .	22
4.3.1.5	vglGreenDiffDisparity . . . . .	22
4.3.1.6	vglHomography . . . . .	22
4.3.1.7	vglMapTo3D . . . . .	22
4.3.1.8	vglMeanMipmap . . . . .	23
4.3.1.9	vglMeanSq3 . . . . .	23
4.3.1.10	vglRectify . . . . .	23
4.3.1.11	vglSumDiff . . . . .	23
4.3.1.12	vglSumDiffMipmap . . . . .	23
4.3.1.13	vglUndistort . . . . .	24
4.4	src/vglImage.cpp File Reference . . . . .	24
4.4.1	Function Documentation . . . . .	25
4.4.1.1	iplPrintImageInfo . . . . .	25
4.4.1.2	LoadPGM . . . . .	25
4.4.1.3	SavePGM . . . . .	25
4.4.1.4	SavePPM . . . . .	26
4.4.1.5	SaveYUV411 . . . . .	26
4.4.1.6	vglBaricenterVga . . . . .	26
4.4.1.7	vglCErodeCross3 . . . . .	26
4.4.1.8	vglClear . . . . .	26
4.4.1.9	vglCloseSq3 . . . . .	26
4.4.1.10	vglCopyCreateImage . . . . .	27
4.4.1.11	vglCopyCreateImage . . . . .	27
4.4.1.12	vglCopyImageTex . . . . .	27
4.4.1.13	vglCopyImageTexFS . . . . .	27
4.4.1.14	vglCopyImageTexVFS . . . . .	27
4.4.1.15	vglCreateImage . . . . .	27
4.4.1.16	vglCreateImage . . . . .	27
4.4.1.17	vglCreateImage . . . . .	27
4.4.1.18	vglDistTransform5 . . . . .	28

4.4.1.19	vglDistTransformCross3 . . . . .	28
4.4.1.20	vglDistTransformSq3 . . . . .	28
4.4.1.21	vglDownload . . . . .	28
4.4.1.22	vglDownloadFaster . . . . .	28
4.4.1.23	vglDownloadFBO . . . . .	29
4.4.1.24	vglDownloadPGM . . . . .	29
4.4.1.25	vglDownloadPPM . . . . .	29
4.4.1.26	vglErodeSq3Sep . . . . .	29
4.4.1.27	vglErodeSq5Sep . . . . .	29
4.4.1.28	vglGetLevelDistTransform5 . . . . .	30
4.4.1.29	vglHorizontalFlip2 . . . . .	30
4.4.1.30	vglInit . . . . .	30
4.4.1.31	vglInit . . . . .	30
4.4.1.32	vglInOut_model . . . . .	30
4.4.1.33	vglLoadImage . . . . .	30
4.4.1.34	vglLoadPGM . . . . .	30
4.4.1.35	vglMultiInput_model . . . . .	31
4.4.1.36	vglMultiOutput_model . . . . .	31
4.4.1.37	vglOpenSq3 . . . . .	31
4.4.1.38	vglPrintImageInfo . . . . .	31
4.4.1.39	vglReleaseImage . . . . .	31
4.4.1.40	vglReplaceIpl . . . . .	31
4.4.1.41	vglSavePGM . . . . .	31
4.4.1.42	vglSavePPM . . . . .	32
4.4.1.43	vglThinBernard . . . . .	32
4.4.1.44	vglThinChin . . . . .	32
4.4.1.45	vglUpload . . . . .	32
4.4.1.46	vglVerticalFlip2 . . . . .	33





## Chapter 1

# Directory Hierarchy

### 1.1 Directories

This directory hierarchy is sorted roughly, but not completely, alphabetically:

src . . . . . 5



# Chapter 2

## File Index

### 2.1 File List

Here is a list of all files with brief descriptions:

src/gls12cpp_BG.h . . . . .	7
src/gls12cpp_shaders.h . . . . .	8
src/gls12cpp_Stereo.h . . . . .	21
src/vgllImage.cpp . . . . .	24



## Chapter 3

# Directory Documentation

### 3.1 src/ Directory Reference

#### Files

- file `glsl2cpp_BG.h`
- file `glsl2cpp_shaders.h`
- file `glsl2cpp_Stereo.h`
- file `vglImage.cpp`



## Chapter 4

# File Documentation

### 4.1 src/gls12cpp\_BG.h File Reference

```
#include "vglImage.h"
```

#### Functions

- void **vglDetectFGSimpleBGModel** (VgllImage \*img\_in, VgllImage \*average, VgllImage \*variance, VgllImage \*foreground, float std\_thresh)
- void **vglTrainSimpleBGModel** (VgllImage \*img\_in, VgllImage \*average, VgllImage \*variance, float weight)
- void **vglUpdatePartialSimpleBGModel** (VgllImage \*img\_in, VgllImage \*foregroundClose, VgllImage \*average, VgllImage \*variance, float weight)

#### 4.1.1 Function Documentation

4.1.1.1 void **vglDetectFGSimpleBGModel** ( VgllImage \* *img\_in*, VgllImage \* *average*, VgllImage \* *variance*, VgllImage \* *foreground*, float *std\_thresh* )

Detects foreground pixels.

4.1.1.2 void **vglTrainSimpleBGModel** ( VgllImage \* *img\_in*, VgllImage \* *average*, VgllImage \* *variance*, float *weight* )

Updates average and variance of background model.

4.1.1.3 `void vglUpdatePartialSimpleBGModel ( VgllImage * img_in, VgllImage * foregorundClose, VgllImage * average, VgllImage * variance, float weight )`

Updates average and variance of background model only in pixels that are classified as background.

## 4.2 `src/gsl2cpp_shaders.h` File Reference

```
#include "vglImage.h"
```

### Functions

- `void shader_15_1 (VgllImage *src, VgllImage *dst)`
- `void vgl1to3Channels (VgllImage *src, VgllImage *dst)`
- `void vglAbsDiff (VgllImage *src0, VgllImage *src1, VgllImage *dst)`
- `void vglAnd (VgllImage *src0, VgllImage *src1, VgllImage *dst)`
- `void vglBaricenterInit (VgllImage *src, VgllImage *dst)`
- `void vglBlurSq3 (VgllImage *src, VgllImage *dst)`
- `void vglClear2 (VgllImage *src_dst, float r, float g, float b, float a=0.0)`
- `void vglContrast (VgllImage *src, VgllImage *dst, float factor)`
- `void vglCoordToColor (VgllImage *dst)`
- `void vglCopy (VgllImage *src, VgllImage *dst)`
- `void vglCrossingNumber (VgllImage *src, VgllImage *dst)`
- `void vglDeleteSkeletonCorners (VgllImage *src, VgllImage *dst, int step)`
- `void vglDeleteSkeletonWarts (VgllImage *src, VgllImage *dst)`
- `void vglDeleteSkeletonWarts2 (VgllImage *src, VgllImage *dst)`
- `void vglDiff (VgllImage *src0, VgllImage *src1, VgllImage *dst)`
- `void vglDilateCross3 (VgllImage *src, VgllImage *dst)`
- `void vglDilateSq3 (VgllImage *src, VgllImage *dst)`
- `void vglErodeCross3 (VgllImage *src, VgllImage *dst)`
- `void vglErodeHL3 (VgllImage *src, VgllImage *dst)`
- `void vglErodeHL5 (VgllImage *src, VgllImage *dst)`
- `void vglErodeHL7 (VgllImage *src, VgllImage *dst)`
- `void vglErodeSq3 (VgllImage *src, VgllImage *dst)`
- `void vglErodeSq3off (VgllImage *src, VgllImage *dst)`
- `void vglErodeSq5 (VgllImage *src, VgllImage *dst)`
- `void vglErodeSq5off (VgllImage *src, VgllImage *dst)`
- `void vglErodeSq7 (VgllImage *src, VgllImage *dst)`
- `void vglErodeSqSide (VgllImage *src, VgllImage *dst, int side)`
- `void vglErodeVL3 (VgllImage *src, VgllImage *dst)`
- `void vglErodeVL5 (VgllImage *src, VgllImage *dst)`
- `void vglErodeVL7 (VgllImage *src, VgllImage *dst)`
- `void vglFeaturePoints (VgllImage *src, VgllImage *dst, int type)`
- `void vglGaussianBlurSq3 (VgllImage *src, VgllImage *dst)`
- `void vglGray (VgllImage *src, VgllImage *dst)`



- void **vglHorizontalFlip** (VgllImage \*src, VgllImage \*dst)
- void **vglInOut** (VgllImage \*src, VgllImage \*dst)
- void **vglJulia** (VgllImage \*dst, float ox=0.0, float oy=0.0, float half\_win=1.0, float c\_real=-1.36, float c\_imag=.11)
- void **vglLaplaceSq3** (VgllImage \*src, VgllImage \*dst)
- void **vglMandel** (VgllImage \*dst, float ox=0.0, float oy=0.0, float half\_win=1.0)
- void **vglMipmap** (VgllImage \*src, VgllImage \*dst, float lod)
- void **vglMulScalar** (VgllImage \*src, VgllImage \*dst, float factor)
- void **vglMultiInput** (VgllImage \*src0, VgllImage \*src1, VgllImage \*dst, float weight=.5)
- void **vglMultiOutput** (VgllImage \*src, VgllImage \*dst, VgllImage \*dst1)
- void **vglNoise** (VgllImage \*src, VgllImage \*dst)
- void **vglNot** (VgllImage \*src, VgllImage \*dst)
- void **vglOr** (VgllImage \*src0, VgllImage \*src1, VgllImage \*dst)
- void **vglRescale** (VgllImage \*src, VgllImage \*dst, float x0, float y0, float x1, float y1)
- void **vglRgbToBgr** (VgllImage \*src, VgllImage \*dst)
- void **vglRgbToHsl** (VgllImage \*src, VgllImage \*dst)
- void **vglRgbToHsv** (VgllImage \*src, VgllImage \*dst)
- void **vglRgbToXYZ** (VgllImage \*src, VgllImage \*dst)
- void **vglRobertsGradient** (VgllImage \*src, VgllImage \*dst)
- void **vglSelfSum22** (VgllImage \*src, VgllImage \*dst)
- void **vglSelfSum3v** (VgllImage \*src, VgllImage \*dst)
- void **vglSelfSum4h** (VgllImage \*src, VgllImage \*dst)
- void **vglSelfSum5h** (VgllImage \*src, VgllImage \*dst)
- void **vglSelfSum5v** (VgllImage \*src, VgllImage \*dst)
- void **vglSharpenSq3** (VgllImage \*src, VgllImage \*dst)
- void **vglSobelGradient** (VgllImage \*src, VgllImage \*dst)
- void **vglSobelXSq3** (VgllImage \*src, VgllImage \*dst)
- void **vglSobelYSq3** (VgllImage \*src, VgllImage \*dst)
- void **vglSum** (VgllImage \*src0, VgllImage \*src1, VgllImage \*dst)
- void **vglSumWeighted** (VgllImage \*src0, VgllImage \*src1, VgllImage \*dst, float weight=.5)
- void **vglSwapRGB** (VgllImage \*src, VgllImage \*dst)
- void **vglTestInOut** (VgllImage \*src\_dst, float r, float g, float b, float a=0.0)
- void **vglTestInOut2** (VgllImage \*src\_dst, VgllImage \*dst)
- void **vglTestMultiInput** (VgllImage \*src0, VgllImage \*src1, VgllImage \*dst, float weight=.5)
- void **vglTestMultiOutput** (VgllImage \*src, VgllImage \*dst, VgllImage \*dst1)
- void **vglTeste** (VgllImage \*src, VgllImage \*dst)
- void **vglThinBernardAux** (VgllImage \*src, VgllImage \*eroded, VgllImage \*dst)
- void **vglThinChinAux** (VgllImage \*src, VgllImage \*dst)
- void **vglThresh** (VgllImage \*src, VgllImage \*dst, float thresh, float top=1.0)
- void **vglThreshLevelSet** (VgllImage \*src, VgllImage \*dst, float thresh, float top=1.0)
- void **vglVerticalFlip** (VgllImage \*src, VgllImage \*dst)
- void **vglWhiteRohrerEdge** (VgllImage \*src, VgllImage \*dst, float radius)
- void **vglXGY** (VgllImage \*src, VgllImage \*dst)
- void **vglZoom** (VgllImage \*src, VgllImage \*dst, float factor)

## 4.2.1 Function Documentation

4.2.1.1 void `shader_15_1` ( *VgllImage \* src*, *VgllImage \* dst* )

4.2.1.2 void `vgl1to3Channels` ( *VgllImage \* src*, *VgllImage \* dst* )

Convert grayscale image to RGB

4.2.1.3 void `vglAbsDiff` ( *VgllImage \* src0*, *VgllImage \* src1*, *VgllImage \* dst* )

Absolute difference between two images.

Referenced by `vglGetLevelDistTransform5()`.

4.2.1.4 void `vglAnd` ( *VgllImage \* src0*, *VgllImage \* src1*, *VgllImage \* dst* )

Logical AND between two images

4.2.1.5 void `vglBaricenterInit` ( *VgllImage \* src*, *VgllImage \* dst* )

Initialize image to be used in baricenter calculation. The initialization is done by storing the values (1, x, y) in each output pixel so that the summation over the whole image gives the three moments of the image.

$R = f(x, y)$

$G = x * f(x, y)$

$B = y * f(x, y)$

Referenced by `vglBaricenterVga()`.

4.2.1.6 void `vglBlurSq3` ( *VgllImage \* src*, *VgllImage \* dst* )

`vglBlurSq3`

Blur image by 3x3 square structuring element.

4.2.1.7 void `vglClear2` ( *VgllImage \* src\_dst*, float *r*, float *g*, float *b*, float *a* = 0.0 )

Clear image with given color.

4.2.1.8 void `vglContrast` ( *VgllImage \* src*, *VgllImage \* dst*, float *factor* )

Changes contrast of image by given factor.

**4.2.1.9 void vglCoordToColor ( Vgllmage \* *dst* )**

Shows coordinates of pixels as colors. Red is horizontal and green is vertical. - Coordinates and colors are defined by OpenGL, that is, between 0 and 1.

**4.2.1.10 void vglCopy ( Vgllmage \* *src*, Vgllmage \* *dst* )**

Direct copy from *src* to *dst*.

Referenced by vglCopyCreateImage(), vglDistTransform5(), vglGetLevelDistTransform5(), vglThinBernard(), and vglThinChin().

**4.2.1.11 void vglCrossingNumber ( Vgllmage \* *src*, Vgllmage \* *dst* )**

Crossing number is defined as the number of occurrences of the pattern 01 in the neighborhood of a pixel.

Neighborhood of pixel *P* is indexed as follows:

```

P3  P2  P1
P4  P  P0/8
P5  P6  P7

```

References:

M. Couprie, Note on fifteen 2D parallel thinning algorithms, 2006

T. M. Bernard and A. Manzanera, Improved low complexity fully parallel thinning algorithms, 1999

**4.2.1.12 void vglDeleteSkeletonCorners ( Vgllmage \* *src*, Vgllmage \* *dst*, int *step* )**

Deletes corner from skeleton.

Receive as input the image with the skeleton to be thinned. Receives also the step. must be called once with step 1 and once with step 2.

Neighborhood pixels is indexed as follows:

```

P3  P2  P1
P4  P8  P0
P5  P6  P7

```

Pixels deleted are the ones that mach the pattern and its rotations by 90deg.

```

0  0  x
0  1  1
x  1  0

```

References:

M. Couprie, Note on fifteen 2D parallel thinning algorithms, 2006

T. M. Bernard and A. Manzanera, Improved low complexity fully parallel thinning algorithms, 1999

#### 4.2.1.13 void **vgDeleteSkeletonWarts** ( VgImage \* *src*, VgImage \* *dst* )

Deletes warts from skeleton. Receive as input the image with the skeleton to be thinned. Neighborhood pixels are indexed as follows:

```

P3 P2 P1
P4 P P0/8
P5 P6 P7

```

Pixels deleted are the ones that mach the pattern and its rotations by 45deg.

```

1 0 0
1 1 0
1 0 0

```

That is the same as delete the pixels with crossing number = 1 and neighbor number = 3

References:

Ke Liu et al., Identification of fork points on the skeletons of handwritten chinese characters

#### 4.2.1.14 void **vgDeleteSkeletonWarts2**( VgImage \* *src*, VgImage \* *dst* )

Deletes warts from skeleton. Receive as input the image with the skeleton to be thinned. Neighborhood pixels are indexed as follows:

P3 P2 P1

P4 P P0/8

P5 P6 P7

Pixels deleted are the ones that mach the pattern and its rotations by 45deg.

```

1 0 0 1 1 0 1 0 0
1 1 0 1 1 0 1 0 0
1 1 0 1 1 0 1 1 0
1 1 1 1 1 0 1 1 0
1 1 1 1 1 0 1 1 1

```

That is the same as delete the pixels with crossing number = 1 and neighbor number  $\geq 3$

References:

Ke Liu et al., Identification of fork points on the skeletons of handwritten chinese characters

#### 4.2.1.15 void **vgDiff** ( VgImage \* *src0*, VgImage \* *src1*, VgImage \* *dst* )

Image *src0* minus *src1*.

**4.2.1.16 void vglDilateCross3 ( VgllImage \* *src*, VgllImage \* *dst* )**

Dilation of image by 3x3 cross structuring element.

**4.2.1.17 void vglDilateSq3 ( VgllImage \* *src*, VgllImage \* *dst* )**

Dilation of image by 3x3 square structuring element.

Referenced by vglCloseSq3(), and vglOpenSq3().

**4.2.1.18 void vglErodeCross3 ( VgllImage \* *src*, VgllImage \* *dst* )**

Erosion of image by 3x3 cross structuring element.

Referenced by vglCErodeCross3(), vglDistTransform5(), vglDistTransformCross3(), vglGetLevelDistTransform5(), and vglThinBernard().

**4.2.1.19 void vglErodeHL3 ( VgllImage \* *src*, VgllImage \* *dst* )**

Erosion of image by horizontal line with 3 pixels.

Referenced by vglErodeSq3Sep().

**4.2.1.20 void vglErodeHL5 ( VgllImage \* *src*, VgllImage \* *dst* )**

Erosion of image by horizontal line with 5 pixels.

Referenced by vglErodeSq5Sep().

**4.2.1.21 void vglErodeHL7 ( VgllImage \* *src*, VgllImage \* *dst* )**

Erosion of image by horizontal line with 7 pixels.

**4.2.1.22 void vglErodeSq3 ( VgllImage \* *src*, VgllImage \* *dst* )**

Erosion of image by 3x3 square structuring element.

Referenced by vglCloseSq3(), vglDistTransform5(), vglDistTransformSq3(), vglGetLevelDistTransform5(), and vglOpenSq3().

**4.2.1.23 void vglErodeSq3off ( VgllImage \* *src*, VgllImage \* *dst* )**

Erosion of image by 3x3 square structuring element. Uses an offset array with 9 elements. Slower than vglErodeSq3.

#### 4.2.1.24 void **vglErodeSq5** ( *VgllImage \* src*, *VgllImage \* dst* )

Erosion of image by 5x5 square structuring element.

#### 4.2.1.25 void **vglErodeSq5off** ( *VgllImage \* src*, *VgllImage \* dst* )

Erosion of image by 3x3 square structuring element. Uses an offset array with 25 elements. Slower than **vglErodeSq5**.

#### 4.2.1.26 void **vglErodeSq7** ( *VgllImage \* src*, *VgllImage \* dst* )

Erosion of image by 7x7 square structuring element.

#### 4.2.1.27 void **vglErodeSqSide** ( *VgllImage \* src*, *VgllImage \* dst*, *int side* )

Erosion of image by square structuring element. The parameter "side" is the dimension of the square side in pixels.

#### 4.2.1.28 void **vglErodeVL3** ( *VgllImage \* src*, *VgllImage \* dst* )

Erosion of image by vertical line with 3 pixels.

Referenced by **vglErodeSq3Sep()**.

#### 4.2.1.29 void **vglErodeVL5** ( *VgllImage \* src*, *VgllImage \* dst* )

Erosion of image by vertical line with 5 pixels.

Referenced by **vglErodeSq5Sep()**.

#### 4.2.1.30 void **vglErodeVL7** ( *VgllImage \* src*, *VgllImage \* dst* )

Erosion of image by vertical line with 7 pixels.

#### 4.2.1.31 void **vglFeaturePoints** ( *VgllImage \* src*, *VgllImage \* dst*, *int type* )

Feature Points are defined as function of the crossing number and number of neighbors of a pixel.

The number of neighbors is indicated as Nb. Crossing number is defined as

Nc = number of occurrences of the pattern 01 in the neighborhood of P

Neighborhood pixels are indexed as follows:

P3 P2 P1

P4 P P0

P5 P6 P7

All the ending points are feature points. Are defined as  $Se = \{ P \mid Nc(P) = 1 \}$

Feature points type 1, denoted as S1, are defined as  $S1 = \{ P \mid Nc(P) \geq 3 \}$

Feature points type 2, denoted as S2, are defined as  $S1 = \{ P \mid Nb(P) \geq 3 \}$

Feature points type 3, denoted as S3, are defined as  $S3 = \{ P \mid Nc(P) \geq 3 \text{ or } Nb(P) \geq 4 \}$

References:

Ke Liu et al., Identification of fork points on the skeletons of handwritten chinese characters

**4.2.1.32** void **vglGaussianBlurSq3**( VgllImage \* *src*, VgllImage \* *dst* )

Blurs image by 3x3 square gaussian structuring element.

**4.2.1.33** void **vglGray**( VgllImage \* *src*, VgllImage \* *dst* )

Convert image to grayscale by calculating the scalar product of (r, g, b) and (.2125, .7154, .0721).

**4.2.1.34** void **vglHorizontalFlip**( VgllImage \* *src*, VgllImage \* *dst* )

Flip image horizontally i.e. left becomes right.

Image flip done by shader.

**4.2.1.35** void **vglInOut**( VgllImage \* *src*, VgllImage \* *dst* )

vglInOut

Test and model for IN\_OUT semantics

**4.2.1.36** void **vglJulia**( VgllImage \* *dst*, float *ox* = 0.0, float *oy* = 0.0, float *half\_win* = 1.0, float *c\_real* = -1.36, float *c\_imag* = .11 )

Calculate Julia set

**4.2.1.37** void **vglLaplaceSq3**( VgllImage \* *src*, VgllImage \* *dst* )

Laplacian of image by 3x3 square structuring element.

**4.2.1.38** void **vglMandel** ( *VgllImage \* dst*, float *ox* = 0 . 0, float *oy* = 0 . 0, float *half\_win* = 1 . 0 )

Calculate Mandelbrot set

**4.2.1.39** void **vglMipmap** ( *VgllImage \* src*, *VgllImage \* dst*, float *lod* )

Get specified level of detail.

**4.2.1.40** void **vglMulScalar** ( *VgllImage \* src*, *VgllImage \* dst*, float *factor* )

Multiply image by scalar.

**4.2.1.41** void **vglMultiInput** ( *VgllImage \* src0*, *VgllImage \* src1*, *VgllImage \* dst*, float *weight* = . 5 )

VglAdd

Sum of two images.

**4.2.1.42** void **vglMultiOutput** ( *VgllImage \* src*, *VgllImage \* dst*, *VgllImage \* dst1* )

vglGray

Convert image to grayscale

**4.2.1.43** void **vglNoise** ( *VgllImage \* src*, *VgllImage \* dst* )

Add gaussian noise to image

**4.2.1.44** void **vglNot** ( *VgllImage \* src*, *VgllImage \* dst* )

Inverts image.

**4.2.1.45** void **vglOr** ( *VgllImage \* src0*, *VgllImage \* src1*, *VgllImage \* dst* )

Logical OR between two images

Referenced by vglCErodeCross3().

**4.2.1.46** void **vglRescale** ( *VgllImage \* src*, *VgllImage \* dst*, float *x0*, float *y0*, float *x1*, float *y1* )

Rescales corners of image to given corners



4.2.1.47 void `vglRgbToBgr` ( `VgImage * src`, `VgImage * dst` )

Converts image RGB to BGR color space

4.2.1.48 void `vglRgbToHsl` ( `VgImage * src`, `VgImage * dst` )

Converts image RGB to HSL color space

4.2.1.49 void `vglRgbToHsv` ( `VgImage * src`, `VgImage * dst` )

Converts image RGB to HSV color space

4.2.1.50 void `vglRgbToXyz` ( `VgImage * src`, `VgImage * dst` )

Converts image RGB to XYZ color space.

4.2.1.51 void `vglRobertsGradient` ( `VgImage * src`, `VgImage * dst` )

Roberts gradient of image

4.2.1.52 void `vglSelfSum22` ( `VgImage * src`, `VgImage * dst` )

Stores in output pixel the sum of 4 adjacent pixels of the input image. The width and height of the output image must be half of the input image.

Referenced by `vglBaricenterVga()`.

4.2.1.53 void `vglSelfSum3v` ( `VgImage * src`, `VgImage * dst` )

Stores in output pixel the sum of 3 adjacent pixels of the input image. The height of the output image must be 1/3th of the input image.

Referenced by `vglBaricenterVga()`.

4.2.1.54 void `vglSelfSum4h` ( `VgImage * src`, `VgImage * dst` )

Stores in output pixel the sum of 4 adjacent pixels of the input image. The width of the output image must be 1/4th of the input image.

Referenced by `vglBaricenterVga()`.

4.2.1.55 void `vglSelfSum5h` ( `VgImage * src`, `VgImage * dst` )

Stores in output pixel the sum of 5 adjacent pixels of the input image. The width of the output image must be 1/5th of the input image.

Referenced by `vglBaricenterVga()`.

**4.2.1.56** `void vglSelfSum5v ( VgllImage * src, VgllImage * dst )`

Stores in output pixel the sum of 5 adjacent pixels of the input image. The height of the output image must be 1/5th of the input image.

Referenced by `vglBaricenterVga()`.

**4.2.1.57** `void vglSharpenSq3 ( VgllImage * src, VgllImage * dst )`

Sharpens image using 3x3 square window.

**4.2.1.58** `void vglSobelGradient ( VgllImage * src, VgllImage * dst )`

Sobel gradient of image

**4.2.1.59** `void vglSobelXSq3 ( VgllImage * src, VgllImage * dst )`

Sobel edge filtering in X direction.

**4.2.1.60** `void vglSobelYSq3 ( VgllImage * src, VgllImage * dst )`

Sobel edge filtering in Y direction.

**4.2.1.61** `void vglSum ( VgllImage * src0, VgllImage * src1, VgllImage * dst )`

Sum of two images.

Referenced by `vglDistTransform5()`, `vglDistTransformCross3()`, and `vglDistTransformSq3()`.

**4.2.1.62** `void vglSumWeighted ( VgllImage * src0, VgllImage * src1, VgllImage * dst, float weight = .5 )`

Weighted sum of two images. The first image is multiplied by weight, and the second, by 1 - weight. Default weight is 0.5.

**4.2.1.63** `void vglSwapRGB ( VgllImage * src, VgllImage * dst )`

Convert image from RGB to BGR and vice versa.

4.2.1.64 void `vglTeste` ( `VgllImage * src`, `VgllImage * dst` )

`vglDilate`

Dilation of image by 3x3 square structuring element.

4.2.1.65 void `vglTestInOut` ( `VgllImage * src_dst`, float *r*, float *g*, float *b*, float *a* = 0.0 )

Test and model for IN\_OUT semantics

4.2.1.66 void `vglTestInOut2` ( `VgllImage * src_dst`, `VgllImage * dst` )

Test and model for IN\_OUT semantics, with double output.

4.2.1.67 void `vglTestMultiInput` ( `VgllImage * src0`, `VgllImage * src1`, `VgllImage * dst`, float *weight* = .5 )

Test and model for multiple input functions.

4.2.1.68 void `vglTestMultiOutput` ( `VgllImage * src`, `VgllImage * dst`, `VgllImage * dst1` )

Test and model for multiple output functions.

4.2.1.69 void `vglThinBernardAux` ( `VgllImage * src`, `VgllImage * eroded`, `VgllImage * dst` )

Return one step of thinning. Algorithm by Bernard and Manzanera 1999. Receive as input the image to be thinned and its erosion by a elementary cross structuring element. Neighborhood pixels are indexed as follows:

```

P3  P2  P1
P4  P8  P0
P5  P6  P7

```

References:

M. Couprie, Note on fifteen 2D parallel thinning algorithms, 2006

T. M. Bernard and A. Manzanera, Improved low complexity fully parallel thinning algorithms, 1999

Referenced by `vglThinBernard()`.

4.2.1.70 void `vglThinChinAux` ( `VgllImage * src`, `VgllImage * dst` )

Return one step of thinning. Algorithm by Chin, Wan Stover and Iverson, 1987. - Receive as input the image to be thinned, buffer image and number of times to iterate. Neighborhood pixels are indexed as follows:

<i>x</i>	<i>x</i>	<i>P10</i>	<i>x</i>	<i>x</i>
<i>x</i>	<i>P3</i>	<i>P2</i>	<i>P1</i>	<i>x</i>
<i>P11</i>	<i>P4</i>	<i>P0</i>	<i>P8</i>	<i>P9</i>
<i>x</i>	<i>P5</i>	<i>P6</i>	<i>P7</i>	<i>x</i>
<i>x</i>	<i>x</i>	<i>P12</i>	<i>x</i>	<i>x</i>

References:

M. Couprie, Note on fifteen 2D parallel thinning algorithms, 2006

R. T. Chin et al., A one-pass thinning algorithm and its parallel implementation, 1987

Referenced by `vglThinChin()`.

**4.2.1.71** `void vglThresh ( VgllImage * src, VgllImage * dst, float thresh, float top = 1.0 )`

Threshold of image. If value is greater than threshold, output is top, else, output is 0. Default top value is 1.

Referenced by `vglDistTransform5()`, `vglDistTransformCross3()`, and `vglDistTransformSq3()`.

**4.2.1.72** `void vglThreshLevelSet ( VgllImage * src, VgllImage * dst, float thresh, float top = 1.0 )`

Threshold of image. If value is equal to level, output is top, else, output is 0. Default top value is 1. Use after some Distance Transform to get a single distance level set.

**4.2.1.73** `void vglVerticalFlip ( VgllImage * src, VgllImage * dst )`

Flip image vertically i.e. top becomes bottom.

Image flip done by shader.

**4.2.1.74** `void vglWhiteRohrerEdge ( VgllImage * src, VgllImage * dst, float radius )`

Finds edge by using a White-Rohrer mask.

**4.2.1.75** `void vglXGY ( VgllImage * src, VgllImage * dst )`

Stores sobel edge filtering in X direction in red channel grayscale in y and sobel edge filtering in Y direction in green channel

**4.2.1.76** `void vglZoom ( VgllImage * src, VgllImage * dst, float factor )`

Zoom image by factor.

## 4.3 src/gsl2cpp\_Stereo.h File Reference

```
#include "vglImage.h"
```

### Functions

- void **vglAbsDiffDisparity** (VgllImage \*img\_ref, VgllImage \*img\_2, VgllImage \*dst, float disparity)
- void **vglAbsDiffDisparityMipmap** (VgllImage \*img\_ref, VgllImage \*img\_2, VgllImage \*dst, float disparity, float max\_lod)
- void **vglFindDisparity** (VgllImage \*img\_dif, VgllImage \*img\_disp, float disparity)
- void **vglFindDisparityDiff** (VgllImage \*img\_sum, VgllImage \*img\_disp, VgllImage \*img\_best, float disparity)
- void **vglGreenDiffDisparity** (VgllImage \*img\_ref, VgllImage \*img\_2, VgllImage \*dst, float disparity)
- void **vglHomography** (VgllImage \*img\_src, VgllImage \*img\_dst, float \*f\_homo)
- void **vglMapTo3D** (VgllImage \*img\_map, VgllImage \*img\_3d, float f, float b, float D, float disp\_k=0.0, float h=10.0)
- void **vglMeanMipmap** (VgllImage \*img\_dif, VgllImage \*img\_out, float max\_lod)
- void **vglMeanSq3** (VgllImage \*img\_dif, VgllImage \*img\_out)
- void **vglRectify** (VgllImage \*img\_src, VgllImage \*img\_dst, float \*f\_dist, float \*f\_proj, float \*f\_homo)
- void **vglSumDiff** (VgllImage \*img\_dif, VgllImage \*img\_out)
- void **vglSumDiffMipmap** (VgllImage \*img\_dif, VgllImage \*img\_out, float max\_lod)
- void **vglUndistort** (VgllImage \*img\_src, VgllImage \*img\_dst, float \*f\_dist, float \*f\_proj)

### 4.3.1 Function Documentation

#### 4.3.1.1 void **vglAbsDiffDisparity** ( VgllImage \* *img\_ref*, VgllImage \* *img\_2*, VgllImage \* *dst*, float *disparity* )

Calculate absolute difference between *img\_ref* and *img\_2*. Disparities considered are in the closed interval  $[4*disparity, 4*disparity+3]$ .

The four differences are stored in the RGBA image *dst*.

#### 4.3.1.2 void **vglAbsDiffDisparityMipmap** ( VgllImage \* *img\_ref*, VgllImage \* *img\_2*, VgllImage \* *dst*, float *disparity*, float *max\_lod* )

Calculates average absolute difference between *img\_ref* and *img\_2* at levels of detail in  $[0, max\_lod]$ . Disparities considered are in the closed interval  $[4*disparity, 4*disparity+3]$ .

The four differences are stored in the RGBA image *dst*.

#### 4.3.1.3 void vglFindDisparity ( VgllImage \* *img\_dif*, VgllImage \* *img\_disp*, float *disparity* )

Find best disparity. The first input image, *img\_dif*, contains absolute differences between a pair of images at disparities  $[4*disparity, 4*disparity+3]$ .

The second input image contains the smallest differences found in channel R, and corresponding disparity value in channel A, is also an output image, and is updated whenever a smaller difference is found.

#### 4.3.1.4 void vglFindDisparityDiff ( VgllImage \* *img\_sum*, VgllImage \* *img\_disp*, VgllImage \* *img\_best*, float *disparity* )

Do the same as *vglFindDisparity*, but the smallest difference is stored in *img\_best*, and corresponding disparity in *img\_disp*. Both are input and output images.

#### 4.3.1.5 void vglGreenDiffDisparity ( VgllImage \* *img\_ref*, VgllImage \* *img\_2*, VgllImage \* *dst*, float *disparity* )

Calculate absolute difference between green channel of *img\_ref* and *img\_2*. Disparities considered are in the closed interval  $[4*disparity, 4*disparity+3]$ .

The four differences are stored in the RGBA image *dst*.

#### 4.3.1.6 void vglHomography ( VgllImage \* *img\_src*, VgllImage \* *img\_dst*, float \* *f\_homo* )

Apply homography in *img\_src* and stores result in *img\_dst*.

Important: for matrices the components are written in column major order:

$$\text{mat2 m} = \text{mat2} (1, 2, 3, 4) \Leftrightarrow \text{m} = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$$

In C we build the matrix in line major order, then we must transpose the matrix before using it in OpenGL context.

#### 4.3.1.7 void vglMapTo3D ( VgllImage \* *img\_map*, VgllImage \* *img\_3d*, float *f*, float *b*, float *D*, float *disp\_k* = 0.0, float *h* = 10.0 )

Convert depth map to affine reconstruction

This algorithm ignores the infinite homography.

*img\_map*: input depth map

*img\_3d*: output reconstruction

*f*: focal length in pixels

*b*: baseline in cm

*D*: fixation point or maximum depth

`is_float`: if true, output image will store `z` in cm. If false output image will store `z` as  $255 * (\text{depth} / D)$ . if `depth == D` then `z = 0`.

`disp_k`: If set, single disparity will be used.

`h`: height of camera in cm

**4.3.1.8** `void vglMeanMipmap ( VgllImage * img_dif, VgllImage * img_out, float max_lod )`

Mean of pixel values of levels of detail in `[0, max_lod]`. Result is stored in `img_out`.

**4.3.1.9** `void vglMeanSq3 ( VgllImage * img_dif, VgllImage * img_out )`

Mean filter with a 3x3 square mask.

**4.3.1.10** `void vglRectify ( VgllImage * img_src, VgllImage * img_dst, float * f_dist, float * f_proj, float * f_homo )`

Undistort, correct projection and rectify `img_src` and stores result in `img_dst`, for use with stereo algorithm

The input float array `f_dist` contains the coefficient of radial distortion, and `f_proj` contains the intrinsic parameters of the camera: center of projection (`x` and `y`); focal length in pixels (`x` and `y`). The focal lengths are the same when the pixels are square.

The input float array `f_homo` contains the homography that rectifies the image.

Important: for matrices the components are written in column major order:

$$\text{mat2 } m = \text{mat2 } (1, 2, 3, 4) \Leftrightarrow m = \begin{pmatrix} 1 & 3 \\ 2 & 4 \end{pmatrix}$$

In C we build the matrix in line major order, then we must transpose the matrix before using it in OpenGL context.

**4.3.1.11** `void vglSumDiff ( VgllImage * img_dif, VgllImage * img_out )`

VglSumDiff

Sum of differences

**4.3.1.12** `void vglSumDiffMipmap ( VgllImage * img_dif, VgllImage * img_out, float max_lod )`

VglSumDiffMipmap

Sum of differences

#### 4.3.1.13 void **vglUndistort** ( VgllImage \* *img\_src*, VgllImage \* *img\_dst*, float \* *f\_dist*, float \* *f\_proj* )

Correct camera lens distortion of *img\_src* and stores the result in *img\_dst*.

The input float array *f\_dist* contains the coefficient of radial distortion, and *f\_proj* contains the intrinsic parameters of the camera: center of projection (x and y); focal length in pixels (x and y). The focal lengths are the same when the pixels are square.

Reference:

[http://www.cognotics.com/opencv/docs/1.0/ref/opencvref\\_cv.htm#cv\\_3d](http://www.cognotics.com/opencv/docs/1.0/ref/opencvref_cv.htm#cv_3d)

## 4.4 src/vglImage.cpp File Reference

```
#include <iostream> #include <GL/freeglut_std.h> #include
<GL/freeglut_ext.h> #include <opencv2/highgui/highgui.h>
#include "vglContext.h" #include "vglImage.h" #include
"vglLoadShader.h" #include "glsl2cpp_shaders.h"
```

### Functions

- int **vglInit** ()
- int **vglInit** (int w, int h)
- void **vglUpload** (VgllImage \*image, int swapRGB)
- VgllImage \* **vglCopyCreateImage** (VgllImage \*img\_in)
- VgllImage \* **vglCopyCreateImage** (IplImage \*img\_in, int dim3, int has\_mipmap)
- VgllImage \* **vglCreateImage** (VgllImage \*img\_in)
- VgllImage \* **vglCreateImage** (IplImage \*img\_in, int dim3, int has\_mipmap)
- VgllImage \* **vglCreateImage** (CvSize size, int depth, int nChannels, int dim3, int has\_mipmap)
- void **vglReleaseImage** (VgllImage \*\*p\_image)
- void **vglReplaceIpl** (VgllImage \*image, IplImage \*new\_ipl)
- void **vglDownloadFaster** (VgllImage \*image)
- void **vglDownload** (VgllImage \*image)
- void **vglDownloadFBO** (VgllImage \*image)
- void **vglDownloadPPM** (VgllImage \*image)
- void **vglDownloadPGM** (VgllImage \*image)
- VgllImage \* **vglLoadImage** (char \*filename, int iscolor, int has\_mipmap)
- void **iplPrintImageInfo** (IplImage \*ipl)
- void **vglPrintImageInfo** (VgllImage \*image)
- void **vglCopyImageTex** (VgllImage \*src, VgllImage \*dst)
- void **vglCopyImageTexFS** (VgllImage \*src, VgllImage \*dst)
- void **vglCopyImageTexVFS** (VgllImage \*src, VgllImage \*dst)
- void **vglVerticalFlip2** (VgllImage \*src, VgllImage \*dst)
- void **vglHorizontalFlip2** (VgllImage \*src, VgllImage \*dst)



- void **vglClear** (VgllImage \*image, float r, float g, float b, float a)
- void **vglOpenSq3** (VgllImage \*src, VgllImage \*dst, VgllImage \*buf, int times)
- void **vglCloseSq3** (VgllImage \*src, VgllImage \*dst, VgllImage \*buf, int times)
- void **vglErodeSq3Sep** (VgllImage \*src, VgllImage \*dst, VgllImage \*buf, int times)
- void **vglErodeSq5Sep** (VgllImage \*src, VgllImage \*dst, VgllImage \*buf, int times)
- void **vglCErodeCross3** (VgllImage \*src, VgllImage \*mask, VgllImage \*dst, VgllImage \*buf, int times)
- int **SavePPM** (char \*filename, int w, int h, void \*savebuf)
- int **vglSavePPM** (VgllImage \*img, char \*filename)
- int **SavePGM** (char \*filename, int w, int h, void \*savebuf)
- int **vglSavePGM** (VgllImage \*img, char \*filename)
- IplImage \* **LoadPGM** (char \*filename)
- VgllImage \* **vglLoadPGM** (char \*filename)
- int **SaveYUV411** (char \*filename, int w, int h, void \*savebuf)
- void **vglDistTransformCross3** (VgllImage \*src, VgllImage \*dst, VgllImage \*buf, VgllImage \*buf2, int times)
- void **vglDistTransformSq3** (VgllImage \*src, VgllImage \*dst, VgllImage \*buf, VgllImage \*buf2, int times)
- void **vglDistTransform5** (VgllImage \*src, VgllImage \*dst, VgllImage \*buf, VgllImage \*buf2, int times)
- void **vglGetLevelDistTransform5** (VgllImage \*src, VgllImage \*dst, VgllImage \*buf, VgllImage \*buf2, int times)
- void **vglThinBernard** (VgllImage \*src, VgllImage \*dst, VgllImage \*buf, int times)
- void **vglThinChin** (VgllImage \*src, VgllImage \*dst, VgllImage \*buf, int times)
- void **vglBaricenterVga** (VgllImage \*src, double \*x\_avg, double \*y\_avg, double \*pix\_count)
- void **vglMultiOutput\_model** (VgllImage \*src, VgllImage \*dst, VgllImage \*dst1)
- void **vglInOut\_model** (VgllImage \*dst, VgllImage \*dst1)
- void **vglMultiInput\_model** (VgllImage \*src0, VgllImage \*src1, VgllImage \*dst)

#### 4.4.1 Function Documentation

##### 4.4.1.1 void **iplPrintImageInfo** ( IplImage \* *ipl* )

Print information about image.

Print width, height, depth and number of channels

##### 4.4.1.2 IplImage\* **LoadPGM** ( char \* *filename* )

Load image data from PGM file, 1 channel, unsigned byte

Referenced by vglLoadPGM().

##### 4.4.1.3 int **SavePGM** ( char \* *filename*, int *w*, int *h*, void \* *savebuf* )

Save image data to PGM file, 1 channel, unsigned byte

Referenced by vglSavePGM().

#### 4.4.1.4 int SavePPM ( char \* filename, int w, int h, void \* savebuf )

Save image data to PPM file, 3 channels, unsigned byte

Time to save a VGA image = 3.5ms

Referenced by vglSavePPM().

#### 4.4.1.5 int SaveYUV411 ( char \* filename, int w, int h, void \* savebuf )

Save compressed YUV411 image data to file.

Requires one half of the disk space required to save an uncompressed PPM.

#### 4.4.1.6 void vglBaricenterVga ( VglImage \* src, double \* x\_avg, double \* y\_avg, double \* pix\_count )

Calculates baricenter of vga image.

The result is stored in image RGB with one pixel.  $R = M(0, 0) = \sum f(x, y)$   $G = M(1, 0) = \sum x * f(x, y)$   $B = M(0, 1) = \sum y * f(x, y)$

Reference:

William K. Pratt, Digital Image Processing, Second Edition

References vglBaricenterInit(), vglCreateImage(), vglDownload(), vglSelfSum22(), vglSelfSum3v(), vglSelfSum4h(), vglSelfSum5h(), and vglSelfSum5v().

#### 4.4.1.7 void vglCErodeCross3 ( VglImage \* src, VglImage \* mask, VglImage \* dst, VglImage \* buf, int times )

Morphological conditional erosion by cross structuring element 3x3. A buffer is required. Source and destination may be the same.

The structuring element is a 3x3 cross. The parameter "times" indicates how many times the erosion will be applied.

References vglErodeCross3(), and vglOr().

#### 4.4.1.8 void vglClear ( VglImage \* image, float r, float g, float b, float a )

Referenced by vglDistTransform5(), vglDistTransformCross3(), and vglDistTransformSq3().

#### 4.4.1.9 void vglCloseSq3 ( VglImage \* src, VglImage \* dst, VglImage \* buf, int times )

Morphological closing by square structuring element. A buffer is required. Source and destination may be the same.

The structuring element is a 3x3 square. The parameter "times" indicates how many times the closing will be applied.

References `vglDilateSq3()`, and `vglErodeSq3()`.

#### 4.4.1.10 `VglImage* vglCopyCreateImage ( VglImage * img_in )`

Create image with same format and data as `img_in`.

References `vglCopy()`, and `vglCreateImage()`.

#### 4.4.1.11 `VglImage* vglCopyCreateImage ( lplImage * img_in, int dim3, int has_mipmap )`

Create image with same format and data as `img_in`

References `vglCreateImage()`, and `vglUpload()`.

#### 4.4.1.12 `void vglCopyImageTex ( VglImage * src, VglImage * dst )`

Copy data from `src` texture to `dst` texture

#### 4.4.1.13 `void vglCopyImageTexFS ( VglImage * src, VglImage * dst )`

Copy data from `src` texture to `dst` texture using a fragment shader

#### 4.4.1.14 `void vglCopyImageTexVFS ( VglImage * src, VglImage * dst )`

Copy data from `src` texture to `dst` texture using a fragment shader and a vertex shader

#### 4.4.1.15 `VglImage* vglCreateImage ( VglImage * img_in )`

Create image with same format as `img_in`

Referenced by `vglBaricenterVga()`, `vglCopyCreateImage()`, `vglCreateImage()`, and `vglLoadPGM()`.

#### 4.4.1.16 `VglImage* vglCreateImage ( lplImage * img_in, int dim3, int has_mipmap )`

Create image with same format as `img_in`

References `vglCreateImage()`.

#### 4.4.1.17 `VglImage* vglCreateImage ( CvSize size, int depth, int nChannels, int dim3, int has_mipmap )`

Create image as described by the parameters

References `vglUpload()`.

**4.4.1.18** `void vglDistTransform5 ( VgllImage * src, VgllImage * dst, VgllImage * buf,  
VgllImage * buf2, int times )`

Distance transform given by alternating an elementary cross and a square 3x3.

Perform successive erorions on input image thresholded to 1/256. The sum of the erosions results is returned as the distance transform result.

References `vglClear()`, `vglCopy()`, `vglErodeCross3()`, `vglErodeSq3()`, `vglSum()`, and `vglThresh()`.

**4.4.1.19** `void vglDistTransformCross3 ( VgllImage * src, VgllImage * dst, VgllImage * buf,  
VgllImage * buf2, int times )`

Distance transform given by elementary cross.

Perform successive erorions on input image thresholded to 1/256. The sum of the erosions results is returned as the distance transform result.

References `vglClear()`, `vglErodeCross3()`, `vglSum()`, and `vglThresh()`.

**4.4.1.20** `void vglDistTransformSq3 ( VgllImage * src, VgllImage * dst, VgllImage * buf,  
VgllImage * buf2, int times )`

Distance transform given by square 3x3.

Perform successive erorions on input image thresholded to 1/256. The sum of the erosions results is returned as the distance transform result.

References `vglClear()`, `vglErodeSq3()`, `vglSum()`, and `vglThresh()`.

**4.4.1.21** `void vglDownload ( VgllImage * image )`

Force transfer of image from GPU to RAM. Add RAM as valid context.

Transfer done by `glGetTexImage`. Color order is compatible with `iplImage`, that is, BGR.

Time to transfer a VGA image = 2.5ms

References `vglPrintImageInfo()`.

Referenced by `vglBaricenterVga()`.

**4.4.1.22** `void vglDownloadFaster ( VgllImage * image )`

Force transfer of image from GPU to RAM. Add RAM as valid context.

Transfer done by `glReadPixels`. Color order is compatible with `iplImage`, that is, BGR.

Time to transfer a VGA image = 1.0 to 1.5ms

References vglPrintImageInfo().

#### 4.4.1.23 void vglDownloadFBO ( VglImage \* *image* )

Force transfer of image from FBO to RAM. Add RAM as valid context.

Transfer done by glReadPixels. Color order is compatible with iplImage, that is, BGR.

References vglPrintImageInfo().

#### 4.4.1.24 void vglDownloadPGM ( VglImage \* *image* )

Transfer image from GPU to RAM in format suitable for saving as PGM.

Use it immediately before vglSavePGM. It is different from vglDownload in two points. The unpack alignment is 1 and color is grayscale

Time to transfer a VGA image = 10ms

Referenced by vglSavePGM().

#### 4.4.1.25 void vglDownloadPPM ( VglImage \* *image* )

Transfer image from GPU to RAM in format suitable for saving as PPM.

Use it immediately before vglSavePPM. It is different from vglDownload in two points. The unpack alignment is 1 and color order is RGB

Time to transfer a VGA image = 3ms

Referenced by vglSavePPM().

#### 4.4.1.26 void vglErodeSq3Sep ( VglImage \* *src*, VglImage \* *dst*, VglImage \* *buf*, int *times* )

Morphological erosion by square structuring element 3x3. A buffer is required. Source and destination may be the same.

The structuring element is a 3x3 square. The parameter "times" indicates how many times the erosion will be applied.

References vglErodeHL3(), and vglErodeVL3().

#### 4.4.1.27 void vglErodeSq5Sep ( VglImage \* *src*, VglImage \* *dst*, VglImage \* *buf*, int *times* )

Morphological erosion by square structuring element 5x5. A buffer is required. Source and destination may be the same.

The structuring element is a 5x5 square. The parameter "times" indicates how many times the erosion will be applied.

References vglErodeHL5(), and vglErodeVL5().

**4.4.1.28** void **vglGetLevelDistTransform5**( VgllImage \* *src*, VgllImage \* *dst*, VgllImage \* *buf*, VgllImage \* *buf2*, int *times* )

Get level curve of distance transform5

Perform successive erorions on input image thresholded to 1/256. The returned image is the difference between the results obtained in the iterations "times" and "times" - 1.

References vglAbsDiff(), vglCopy(), vglErodeCross3(), and vglErodeSq3().

**4.4.1.29** void **vglHorizontalFlip2**( VgllImage \* *src*, VgllImage \* *dst* )

Flip image horizontally, that is, left becomes right.

Image flip done by texture mapping, that is, by the fixed pipeline.

**4.4.1.30** int **vglInit**( )

Initialize GLUT and create output window with default size (1280, 960).

Referenced by vglUpload().

**4.4.1.31** int **vglInit**( int *w*, int *h* )

Initialize GLUT and create output window with size (w, h).

**4.4.1.32** void **vglInOut\_model**( VgllImage \* *dst*, VgllImage \* *dst1* )

Test and model for IN\_OUT semantics.

First parameter is input and output. Second parameter is output.

**4.4.1.33** VgllImage\* **vglLoadImage**( char \* *filename*, int *iscolor*, int *has\_mipmap* )

Load image from file to new VgllImage.

This function uses cvLoadImage to read image file.

References vglUpload().

**4.4.1.34** VgllImage\* **vglLoadPGM**( char \* *filename* )

Load image from PGM file, 1 channel, unsigned byte

References LoadPGM(), and vglCreateImage().

**4.4.1.35 void vglMultiInput\_model ( VgllImage \* *src0*, VgllImage \* *src1*, VgllImage \* *dst* )**

Test and model for functions with multiple input images.

First and second parameters are input. Third parameter is output.

**4.4.1.36 void vglMultiOutput\_model ( VgllImage \* *src*, VgllImage \* *dst*, VgllImage \* *dst1* )**

Test and model for functions with multiple output images.

First parameter is input. Second and third parameters are output.

**4.4.1.37 void vglOpenSq3 ( VgllImage \* *src*, VgllImage \* *dst*, VgllImage \* *buf*, int *times* )**

Morphological opening by square structuring element. Opening is an erosion followed by a dilation. A buffer is required. Source and destination may be the same.

The structuring element is a 3x3 square. The parameter "times" indicates how many times the erosion will be applied.

References vglDilateSq3(), and vglErodeSq3().

**4.4.1.38 void vglPrintImageInfo ( VgllImage \* *image* )**

Print information about image.

Print width, height, depth, number of channels, OpenGL texture handler, OpenGL FBO handler, and current valid context (RAM, GPU or FBO).

Referenced by vglDownload(), vglDownloadFaster(), vglDownloadFBO(), and vglUpload().

**4.4.1.39 void vglReleaseImage ( VgllImage \*\* *p\_image* )**

Release memory occupied by image in RAM and GPU

**4.4.1.40 void vglReplaceIpl ( VgllImage \* *image*, IplImage \* *new\_ipl* )**

Replace IplImage, stored inside a VgllImage, with new IplImage.

Both new and old images must have exactly the same properties, dimensions, depth, type etc. Is useful when grabbing frames from a camera.

References vglUpload().

**4.4.1.41 int vglSavePGM ( VgllImage \* *img*, char \* *filename* )**

Save image to PGM file, 1 channel, unsigned byte

References SavePGM(), and vglDownloadPGM().

#### 4.4.1.42 `int vglSavePPM ( VgllImage * img, char * filename )`

Save image to PPM file, 3 channels, unsigned byte

References `SavePPM()`, and `vglDownloadPPM()`.

#### 4.4.1.43 `void vglThinBernard ( VgllImage * src, VgllImage * dst, VgllImage * buf, int times )`

Structuring element thinning. Algorithm by Bernard and Manzanera 1999.

Receive as input the image to be thinned. The second image is an auxiliary image. The third image stores the result. Both the second and third images must have the same size and type as the first input image.

The fourth parameter is the number of iterations.

Reference:

M. Couprie, Note on fifteen 2D parallel thinning algorithms, 2006

T. M. Bernard and A. Manzanera, Improved low complexity fully parallel thinning algorithms, 1999

References `vglCopy()`, `vglErodeCross3()`, and `vglThinBernardAux()`.

#### 4.4.1.44 `void vglThinChin ( VgllImage * src, VgllImage * dst, VgllImage * buf, int times )`

Structuring element thinning. Algorithm by Chin, Wan Stover and Iverson, 1987.

Receive as input the image to be thinned, buffer image and number of times to iterate.

Neighborhood pixels are indexed as follows:

```

P3  P2  P1
P4  P8  P0
P5  P6  P7

```

Reference:

M. Couprie, Note on fifteen 2D parallel thinning algorithms, 2006

R. T. Chin et al., A one-pass thinning algorithm and its parallel implementation, 1987

References `vglCopy()`, and `vglThinChinAux()`.

#### 4.4.1.45 `void vglUpload ( VgllImage * image, int swapRGB )`

Send image data from RAM to GPU. Add GPU as valid context.

If `swapRGB` is true, channels R and B are swapped.

References `vglInit()`, and `vglPrintImageInfo()`.

Referenced by `vglCopyCreateImage()`, `vglCreateImage()`, `vglLoadImage()`, and `vglReplaceIpl()`.



4.4.1.46 void vglVerticalFlip2 ( VgllImage \* *src*, VgllImage \* *dst* )

Flip image vertically, that is, top becomes bottom.

Image flip done by texture mapping, that is, by the fixed pipeline.

# Index

LoadPGM  
    vglImage.cpp, 25

SavePGM  
    vglImage.cpp, 25

SavePPM  
    vglImage.cpp, 25

SaveYUV411  
    vglImage.cpp, 26

gsl2cpp\_BG.h  
    vglDetectFGSimpleBGModel, 7  
    vglTrainSimpleBGModel, 7  
    vglUpdatePartialSimpleBGModel, 7

gsl2cpp\_Stereo.h  
    vglAbsDiffDisparity, 21  
    vglAbsDiffDisparityMipmap, 21  
    vglFindDisparity, 21  
    vglFindDisparityDiff, 22  
    vglGreenDiffDisparity, 22  
    vglHomography, 22  
    vglMapTo3D, 22  
    vglMeanMipmap, 23  
    vglMeanSq3, 23  
    vglRectify, 23  
    vglSumDiff, 23  
    vglSumDiffMipmap, 23  
    vglUndistort, 23

gsl2cpp\_shaders.h  
    shader\_15\_1, 10  
    vgl1to3Channels, 10  
    vglAbsDiff, 10  
    vglAnd, 10  
    vglBaricenterInit, 10  
    vglBlurSq3, 10  
    vglClear2, 10  
    vglContrast, 10  
    vglCoordToColor, 10  
    vglCopy, 11  
    vglCrossingNumber, 11  
    vglDeleteSkeletonCorners, 11  
    vglDeleteSkeletonWarts, 11  
    vglDeleteSkeletonWarts2, 12  
    vglDiff, 12  
    vglDilateCross3, 12  
    vglDilateSq3, 13  
    vglErodeCross3, 13  
    vglErodeHL3, 13  
    vglErodeHL5, 13  
    vglErodeHL7, 13  
    vglErodeSq3, 13  
    vglErodeSq3off, 13  
    vglErodeSq5, 13  
    vglErodeSq5off, 14  
    vglErodeSq7, 14  
    vglErodeSqSide, 14  
    vglErodeVL3, 14  
    vglErodeVL5, 14  
    vglErodeVL7, 14  
    vglFeaturePoints, 14  
    vglGaussianBlurSq3, 15  
    vglGray, 15  
    vglHorizontalFlip, 15  
    vglInOut, 15  
    vglJulia, 15  
    vglLaplaceSq3, 15  
    vglMandel, 15  
    vglMipmap, 16  
    vglMulScalar, 16  
    vglMultiInput, 16  
    vglMultiOutput, 16  
    vglNoise, 16  
    vglNot, 16  
    vglOr, 16  
    vglRescale, 16  
    vglRgbToBgr, 16  
    vglRgbToHsl, 17  
    vglRgbToHsv, 17  
    vglRgbToXyz, 17  
    vglRobertsGradient, 17  
    vglSelfSum22, 17  
    vglSelfSum3v, 17  
    vglSelfSum4h, 17

- vglSelfSum5h, 17
- vglSelfSum5v, 18
- vglSharpenSq3, 18
- vglSobelGradient, 18
- vglSobelXSq3, 18
- vglSobelYSq3, 18
- vglSum, 18
- vglSumWeighted, 18
- vglSwapRGB, 18
- vglTestInOut, 19
- vglTestInOut2, 19
- vglTestMultiInput, 19
- vglTestMultiOutput, 19
- vglTeste, 18
- vglThinBernardAux, 19
- vglThinChinAux, 19
- vglThresh, 20
- vglThreshLevelSet, 20
- vglVerticalFlip, 20
- vglWhiteRohrerEdge, 20
- vglXGY, 20
- vglZoom, 20
- iplPrintImageInfo
  - vgllImage.cpp, 25
- shader\_15\_1
  - gsl2cpp\_shaders.h, 10
- src/ Directory Reference, 5
- src/gsl2cpp\_BG.h, 7
- src/gsl2cpp\_Stereo.h, 21
- src/gsl2cpp\_shaders.h, 8
- src/vgllImage.cpp, 24
- vgl1to3Channels
  - gsl2cpp\_shaders.h, 10
- vglAbsDiff
  - gsl2cpp\_shaders.h, 10
- vglAbsDiffDisparity
  - gsl2cpp\_Stereo.h, 21
- vglAbsDiffDisparityMipmap
  - gsl2cpp\_Stereo.h, 21
- vglAnd
  - gsl2cpp\_shaders.h, 10
- vglBaricenterInit
  - gsl2cpp\_shaders.h, 10
- vglBaricenterVga
  - vgllImage.cpp, 26
- vglBlurSq3
  - gsl2cpp\_shaders.h, 10
- vglCErodeCross3
  - vgllImage.cpp, 26
- vglClear
  - vgllImage.cpp, 26
- vglClear2
  - gsl2cpp\_shaders.h, 10
- vglCloseSq3
  - vgllImage.cpp, 26
- vglContrast
  - gsl2cpp\_shaders.h, 10
- vglCoordToColor
  - gsl2cpp\_shaders.h, 10
- vglCopy
  - gsl2cpp\_shaders.h, 11
- vglCopyCreateImage
  - vgllImage.cpp, 27
- vglCopyImageTex
  - vgllImage.cpp, 27
- vglCopyImageTexFS
  - vgllImage.cpp, 27
- vglCopyImageTexVFS
  - vgllImage.cpp, 27
- vglCreateImage
  - vgllImage.cpp, 27
- vglCrossingNumber
  - gsl2cpp\_shaders.h, 11
- vglDeleteSkeletonCorners
  - gsl2cpp\_shaders.h, 11
- vglDeleteSkeletonWarts
  - gsl2cpp\_shaders.h, 11
- vglDeleteSkeletonWarts2
  - gsl2cpp\_shaders.h, 12
- vglDetectFGSimpleBGModel
  - gsl2cpp\_BG.h, 7
- vglDiff
  - gsl2cpp\_shaders.h, 12
- vglDilateCross3
  - gsl2cpp\_shaders.h, 12
- vglDilateSq3
  - gsl2cpp\_shaders.h, 13
- vglDistTransform5
  - vgllImage.cpp, 28
- vglDistTransformCross3
  - vgllImage.cpp, 28
- vglDistTransformSq3
  - vgllImage.cpp, 28
- vglDownload
  - vgllImage.cpp, 28
- vglDownloadFBO
  - vgllImage.cpp, 29

- vglDownloadFaster
  - vglImage.cpp, 28
- vglDownloadPGM
  - vglImage.cpp, 29
- vglDownloadPPM
  - vglImage.cpp, 29
- vglErodeCross3
  - gsl2cpp\_shaders.h, 13
- vglErodeHL3
  - gsl2cpp\_shaders.h, 13
- vglErodeHL5
  - gsl2cpp\_shaders.h, 13
- vglErodeHL7
  - gsl2cpp\_shaders.h, 13
- vglErodeSq3
  - gsl2cpp\_shaders.h, 13
- vglErodeSq3Sep
  - vglImage.cpp, 29
- vglErodeSq3off
  - gsl2cpp\_shaders.h, 13
- vglErodeSq5
  - gsl2cpp\_shaders.h, 13
- vglErodeSq5Sep
  - vglImage.cpp, 29
- vglErodeSq5off
  - gsl2cpp\_shaders.h, 14
- vglErodeSq7
  - gsl2cpp\_shaders.h, 14
- vglErodeSqSide
  - gsl2cpp\_shaders.h, 14
- vglErodeVL3
  - gsl2cpp\_shaders.h, 14
- vglErodeVL5
  - gsl2cpp\_shaders.h, 14
- vglErodeVL7
  - gsl2cpp\_shaders.h, 14
- vglFeaturePoints
  - gsl2cpp\_shaders.h, 14
- vglFindDisparity
  - gsl2cpp\_Stereo.h, 21
- vglFindDisparityDiff
  - gsl2cpp\_Stereo.h, 22
- vglGaussianBlurSq3
  - gsl2cpp\_shaders.h, 15
- vglGetLevelDistTransform5
  - vglImage.cpp, 29
- vglGray
  - gsl2cpp\_shaders.h, 15
- vglGreenDiffDisparity
  - gsl2cpp\_Stereo.h, 22
- vglHomography
  - gsl2cpp\_Stereo.h, 22
- vglHorizontalFlip
  - gsl2cpp\_shaders.h, 15
- vglHorizontalFlip2
  - vglImage.cpp, 30
- vglImage.cpp
  - LoadPGM, 25
  - SavePGM, 25
  - SavePPM, 25
  - SaveYUV411, 26
  - iplPrintImageInfo, 25
  - vglBaricenterVga, 26
  - vglCErodeCross3, 26
  - vglClear, 26
  - vglCloseSq3, 26
  - vglCopyCreateImage, 27
  - vglCopyImageTex, 27
  - vglCopyImageTexFS, 27
  - vglCopyImageTexVFS, 27
  - vglCreateImage, 27
  - vglDistTransform5, 28
  - vglDistTransformCross3, 28
  - vglDistTransformSq3, 28
  - vglDownload, 28
  - vglDownloadFBO, 29
  - vglDownloadFaster, 28
  - vglDownloadPGM, 29
  - vglDownloadPPM, 29
  - vglErodeSq3Sep, 29
  - vglErodeSq5Sep, 29
  - vglGetLevelDistTransform5, 29
  - vglHorizontalFlip2, 30
  - vglInOut\_model, 30
  - vglInit, 30
  - vglLoadImage, 30
  - vglLoadPGM, 30
  - vglMultiInput\_model, 30
  - vglMultiOutput\_model, 31
  - vglOpenSq3, 31
  - vglPrintImageInfo, 31
  - vglReleaseImage, 31
  - vglReplacelpl, 31
  - vglSavePGM, 31
  - vglSavePPM, 31
  - vglThinBernard, 32
  - vglThinChin, 32
  - vglUpload, 32
  - vglVerticalFlip2, 32
- vglInOut

---

gsl2cpp\_shaders.h, 15  
vglInOut\_model  
    vglImage.cpp, 30  
vglInit  
    vglImage.cpp, 30  
vglJulia  
    gsl2cpp\_shaders.h, 15  
vglLaplaceSq3  
    gsl2cpp\_shaders.h, 15  
vglLoadImage  
    vglImage.cpp, 30  
vglLoadPGM  
    vglImage.cpp, 30  
vglMandel  
    gsl2cpp\_shaders.h, 15  
vglMapTo3D  
    gsl2cpp\_Stereo.h, 22  
vglMeanMipmap  
    gsl2cpp\_Stereo.h, 23  
vglMeanSq3  
    gsl2cpp\_Stereo.h, 23  
vglMipmap  
    gsl2cpp\_shaders.h, 16  
vglMulScalar  
    gsl2cpp\_shaders.h, 16  
vglMultilInput  
    gsl2cpp\_shaders.h, 16  
vglMultilInput\_model  
    vglImage.cpp, 30  
vglMultiOutput  
    gsl2cpp\_shaders.h, 16  
vglMultiOutput\_model  
    vglImage.cpp, 31  
vglNoise  
    gsl2cpp\_shaders.h, 16  
vglNot  
    gsl2cpp\_shaders.h, 16  
vglOpenSq3  
    vglImage.cpp, 31  
vglOr  
    gsl2cpp\_shaders.h, 16  
vglPrintImageInfo  
    vglImage.cpp, 31  
vglRectify  
    gsl2cpp\_Stereo.h, 23  
vglReleaseImage  
    vglImage.cpp, 31  
vglReplacespl  
    vglImage.cpp, 31  
vglRescale  
    gsl2cpp\_shaders.h, 16  
vglRgbToBgr  
    gsl2cpp\_shaders.h, 16  
vglRgbToHsl  
    gsl2cpp\_shaders.h, 17  
vglRgbToHsv  
    gsl2cpp\_shaders.h, 17  
vglRgbToXyz  
    gsl2cpp\_shaders.h, 17  
vglRobertsGradient  
    gsl2cpp\_shaders.h, 17  
vglSavePGM  
    vglImage.cpp, 31  
vglSavePPM  
    vglImage.cpp, 31  
vglSelfSum22  
    gsl2cpp\_shaders.h, 17  
vglSelfSum3v  
    gsl2cpp\_shaders.h, 17  
vglSelfSum4h  
    gsl2cpp\_shaders.h, 17  
vglSelfSum5h  
    gsl2cpp\_shaders.h, 17  
vglSelfSum5v  
    gsl2cpp\_shaders.h, 18  
vglSharpenSq3  
    gsl2cpp\_shaders.h, 18  
vglSobelGradient  
    gsl2cpp\_shaders.h, 18  
vglSobelXSq3  
    gsl2cpp\_shaders.h, 18  
vglSobelYSq3  
    gsl2cpp\_shaders.h, 18  
vglSum  
    gsl2cpp\_shaders.h, 18  
vglSumDiff  
    gsl2cpp\_Stereo.h, 23  
vglSumDiffMipmap  
    gsl2cpp\_Stereo.h, 23  
vglSumWeighted  
    gsl2cpp\_shaders.h, 18  
vglSwapRGB  
    gsl2cpp\_shaders.h, 18  
vglTestInOut  
    gsl2cpp\_shaders.h, 19  
vglTestInOut2  
    gsl2cpp\_shaders.h, 19  
vglTestMultiInput  
    gsl2cpp\_shaders.h, 19  
vglTestMultiOutput

---

---

- gsl2cpp\_shaders.h, 19
- vglTeste
  - gsl2cpp\_shaders.h, 18
- vglThinBernard
  - vglImage.cpp, 32
- vglThinBernardAux
  - gsl2cpp\_shaders.h, 19
- vglThinChin
  - vglImage.cpp, 32
- vglThinChinAux
  - gsl2cpp\_shaders.h, 19
- vglThresh
  - gsl2cpp\_shaders.h, 20
- vglThreshLevelSet
  - gsl2cpp\_shaders.h, 20
- vglTrainSimpleBGModel
  - gsl2cpp\_BG.h, 7
- vglUndistort
  - gsl2cpp\_Stereo.h, 23
- vglUpdatePartialSimpleBGModel
  - gsl2cpp\_BG.h, 7
- vglUpload
  - vglImage.cpp, 32
- vglVerticalFlip
  - gsl2cpp\_shaders.h, 20
- vglVerticalFlip2
  - vglImage.cpp, 32
- vglWhiteRohrerEdge
  - gsl2cpp\_shaders.h, 20
- vglXGY
  - gsl2cpp\_shaders.h, 20
- vglZoom
  - gsl2cpp\_shaders.h, 20