libvisiongl

1.0

Generated by Doxygen 1.8.13

Contents

Chapter 1

File Index

1.1 File List

Here is a list of all files with brief descriptions:

Chapter 2

File Documentation

2.1 src/cl2cpp_BIN.h File Reference

```
#include "vglImage.h"
#include "vglShape.h"
#include "vglStrEl.h"
```

Functions

- void **vglCl3dBinCopy** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglCl3dBinDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dBinDilatePack** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dBinErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dBinErodePack** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dBinMax** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglCl3dBinMin** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglCl3dBinNot** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglCl3dBinRoi** (VglImage ∗img_output, int x0, int y0, int z0, int xf, int yf, int zf)
- void **vglCl3dBinSub** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglCl3dBinSwap** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglCl3dBinThreshold** (VglImage ∗img_input, VglImage ∗img_output, float thresh)
- void **vglCl3dBinToGray** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglClBinConway** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglClBinCopy** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglClBinDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window↩ _size_x, int window_size_y)
- void **vglClBinDilatePack** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)
- void **vglClBinErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window↩ _size_x, int window_size_y)
- void **vglClBinErodePack** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)
- void **vglClBinMax** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)

- void **vglClBinMin** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglClBinNot** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglClBinRoi** (VglImage ∗img_output, int x0, int y0, int xf, int yf)
- void **vglClBinSub** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglClBinSwap** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglClBinThreshold** (VglImage ∗img_input, VglImage ∗img_output, float thresh)
- void **vglClBinToGray** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglClNdBinCopy** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglClNdBinDilate** (VglImage ∗img_input, VglImage ∗img_output, VglStrEl ∗window)
- void **vglClNdBinDilatePack** (VglImage ∗img_input, VglImage ∗img_output, VglStrEl ∗window)
- void **vglClNdBinErode** (VglImage ∗img_input, VglImage ∗img_output, VglStrEl ∗window)
- void **vglClNdBinErodePack** (VglImage ∗img_input, VglImage ∗img_output, VglStrEl ∗window)
- void **vglClNdBinMax** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglClNdBinMin** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglClNdBinNot** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglClNdBinRoi** (VglImage ∗img_output, int ∗p0, int ∗pf)
- void **vglClNdBinSub** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglClNdBinSwap** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglClNdBinThreshold** (VglImage ∗img_input, VglImage ∗img_output, unsigned char thresh)
- void **vglClNdBinToGray** (VglImage ∗img_input, VglImage ∗img_output)

2.1.1 Function Documentation

2.1.1.1 vglCl3dBinCopy()

```
void vglCl3dBinCopy (
            VglImage ∗ img_input,
            VglImage ∗ img_output )
```

Copy of binary image img_input to img_output.

2.1.1.2 vglCl3dBinDilate()

```
void vglCl3dBinDilate (
            VglImage ∗ img_input,
            VglImage ∗ img_output,
            float ∗ convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Dilation of img_input by mask. Result is stored in img_output.

2.1.1.3 vglCl3dBinDilatePack()

```
void vglCl3dBinDilatePack (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Dilation of img_input by mask. Result is stored in img_output.

2.1.1.4 vglCl3dBinErode()

```
void vglCl3dBinErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Dilation of img_input by mask. Result is stored in img_output.

2.1.1.5 vglCl3dBinErodePack()

```
void vglCl3dBinErodePack (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Erosion of img_input by mask. Result is stored in img_output.

2.1.1.6 vglCl3dBinMax()

```
void vglCl3dBinMax (
            VglImage * img_input1,
            VglImage * img_input2,
            VglImage * img_output )
```

Maximum or union between two images.

Maximum or union between img_input1 and img_input2. Result save in img_output.

2.1.1.7 vglCl3dBinMin()

```
void vglCl3dBinMin (
             VglImage * img_input1,
             VglImage * img_input2,
             VglImage * img_output )
```

Minimum or intersection between two images.

Minimum or intersection between img_input1 and img_input2. Result saved in img_output.

2.1.1.8 vglCl3dBinNot()

```
void vglCl3dBinNot (
             VglImage * img_input,
             VglImage * img_output )
```

Negation of binary image img_input. Result is stored in img_output.

2.1.1.9 vglCl3dBinRoi()

```
void vglCl3dBinRoi (
             VglImage * img_output,
             int x0,
             int y0,
             int z0,
             int xf,
             int yf,
             int zf )
```

Generate ROI.

Generate ROI (Region Of Interest). Useful to be used as mask to do intersection with other images.

2.1.1.10 vglCl3dBinSub()

```
void vglCl3dBinSub (
             VglImage * img_input1,
             VglImage * img_input2,
             VglImage * img_output )
```

Subtraction or difference between two binary images.

Subtraction or difference between two binary images. Finds img_input1 minus img_input2 and saves in img_output.

2.1.1.11 vglCl3dBinSwap()

```
void vglCl3dBinSwap (
             VglImage * img_input,
             VglImage * img_output )
```

Negation of binary image img_input. Result is stored in img_output.

2.1.1.12 vglCl3dBinThreshold()

```
void vglCl3dBinThreshold (
            VglImage * img_input,
            VglImage * img_output,
            float thresh )
```

Threshold of grayscale image with binary result.

Threshold of grayscale image img_input. Result is binary, stored in img_output. Parameter thresh is float between 0.0 and 1.0.

2.1.1.13 vglCl3dBinToGray()

```
void vglCl3dBinToGray (
            VglImage * img_input,
            VglImage * img_output )
```

Convert binary image to grayscale.

Convert binary image to grayscale.

2.1.1.14 vglClBinConway()

```
void vglClBinConway (
            VglImage * img_input,
            VglImage * img_output )
```

Conway game of life.

2.1.1.15 vglClBinCopy()

```
void vglClBinCopy (
            VglImage * img_input,
            VglImage * img_output )
```

Copy of binary image img_input to img_output.

2.1.1.16 vglClBinDilate()

```
void vglClBinDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Dilation of img_input by mask. Result is stored in img_output.

2.1.1.17 vglClBinDilatePack()

```
void vglClBinDilatePack (
             VglImage * img_input,
             VglImage * img_output,
             float * convolution_window,
             int window_size_x,
             int window_size_y )
```

Dilation of img_input by mask. Result is stored in img_output.

2.1.1.18 vglClBinErode()

```
void vglClBinErode (
             VglImage * img_input,
             VglImage * img_output,
             float * convolution_window,
             int window_size_x,
             int window_size_y )
```

Erosion of img_input by mask. Result is stored in img_output.

2.1.1.19 vglClBinErodePack()

```
void vglClBinErodePack (
             VglImage * img_input,
             VglImage * img_output,
             float * convolution_window,
             int window_size_x,
             int window_size_y )
```

Dilation of img_input by mask. Result is stored in img_output.

2.1.1.20 vglClBinMax()

```
void vglClBinMax (
             VglImage * img_input1,
             VglImage * img_input2,
             VglImage * img_output )
```

Maximum or union between two images.

Maximum or union between img_input1 and img_input2. Result saved in img_output.

2.1.1.21 vglClBinMin()

```
void vglClBinMin (
             VglImage * img_input1,
             VglImage * img_input2,
             VglImage * img_output )
```

Minimum or intersection between two images.

Minimum or intersection between img_input1 and img_input2. Result saved in img_output.

2.1.1.22 vglClBinNot()

```
void vglClBinNot (
            VglImage * img_input,
            VglImage * img_output )
```

Negation of binary image img_input. Result is stored in img_output.

2.1.1.23 vglClBinRoi()

```
void vglClBinRoi (
            VglImage * img_output,
            int x0,
            int y0,
            int xf,
            int yf )
```

Generate ROI.

Generate ROI (Region Of Interest). Useful to be used as mask to do intersection with other images.

2.1.1.24 vglClBinSub()

```
void vglClBinSub (
            VglImage * img_input1,
            VglImage * img_input2,
            VglImage * img_output )
```

Subtraction or difference between two binary images.

Subtraction or difference between two binary images. Finds img_input1 minus img_input2 and saves in img_output.

2.1.1.25 vglClBinSwap()

```
void vglClBinSwap (
            VglImage * img_input,
            VglImage * img_output )
```

Negation of binary image img_input. Result is stored in img_output.

2.1.1.26 vglClBinThreshold()

```
void vglClBinThreshold (
            VglImage * img_input,
            VglImage * img_output,
            float thresh )
```

Threshold of grayscale image with binary result.

Threshold of grayscale image img_input. Result is binary, stored in img_output. Parameter thresh is float between 0.0 and 1.0.

2.1.1.27 vglClBinToGray()

```
void vglClBinToGray (
             VglImage * img_input,
             VglImage * img_output )
```

Convert binary image to grayscale.

Convert binary image to grayscale.

2.1.1.28 vglClNdBinCopy()

```
void vglClNdBinCopy (
             VglImage * img_input,
             VglImage * img_output )
```

Copy N-dimensional image word by word.

2.1.1.29 vglClNdBinDilate()

```
void vglClNdBinDilate (
             VglImage * img_input,
             VglImage * img_output,
             VglStrEl * window )
```

N-dimensional dilation

SHAPE directive passes a structure with size of each dimension, offsets and number of dimensions. Parameter does not appear in wrapper parameter list. The C expression between parenthesis returns the desired shape of type VglClShape.

2.1.1.30 vglClNdBinDilatePack()

```
void vglClNdBinDilatePack (
             VglImage * img_input,
             VglImage * img_output,
             VglStrEl * window )
```

N-dimensional dilation

SHAPE directive passes a structure with size of each dimension, offsets and number of dimensions. Parameter does not appear in wrapper parameter list. The C expression between parenthesis returns the desired shape of type VglClShape.

2.1.1.31 vglClNdBinErode()

```
void vglClNdBinErode (
             VglImage * img_input,
             VglImage * img_output,
             VglStrEl * window )
```

N-dimensional erosion

SHAPE directive passes a structure with size of each dimension, offsets and number of dimensions. Parameter does not appear in wrapper parameter list. The C expression between parenthesis returns the desired shape of type VglClShape.

2.1.1.32 vglClNdBinErodePack()

```
void vglClNdBinErodePack (
            VglImage * img_input,
            VglImage * img_output,
            VglStrEl * window )
```

N-dimensional erosion

SHAPE directive passes a structure with size of each dimension, offsets and number of dimensions. Parameter does not appear in wrapper parameter list. The C expression between parenthesis returns the desired shape of type VglClShape.

2.1.1.33 vglClNdBinMax()

```
void vglClNdBinMax (
            VglImage * img_input1,
            VglImage * img_input2,
            VglImage * img_output )
```

Maximum or union between two images.

Maximum or union between img_input1 and img_input2. Result saved in img_output.

2.1.1.34 vglClNdBinMin()

```
void vglClNdBinMin (
            VglImage * img_input1,
            VglImage * img_input2,
            VglImage * img_output )
```

Minimum or intersection between two images.

Minimum or intersection between img_input1 and img_input2. Result saved in img_output.

2.1.1.35 vglClNdBinNot()

```
void vglClNdBinNot (
            VglImage * img_input,
            VglImage * img_output )
```

Negation of binary image img_input. Result is stored in img_output.

2.1.1.36 vglClNdBinRoi()

```
void vglClNdBinRoi (
            VglImage * img_output,
            int * p0,
            int * pf )
```

Generate ROI.

Generate ROI (Region Of Interest). Useful to be used as mask to do intersection with other images.

2.1.1.37 vglClNdBinSub()

```
void vglClNdBinSub (
            VglImage ∗ img_input1,
            VglImage ∗ img_input2,
            VglImage ∗ img_output )
```

Subtraction or difference between two binary images.

Subtraction or difference between two binary images. Finds img_input1 minus img_input2 and saves in img_output.

2.1.1.38 vglClNdBinSwap()

```
void vglClNdBinSwap (
            VglImage ∗ img_input,
            VglImage ∗ img_output )
```

Negation of binary image img_input. Result is stored in img_output.

2.1.1.39 vglClNdBinThreshold()

```
void vglClNdBinThreshold (
            VglImage ∗ img_input,
            VglImage ∗ img_output,
            unsigned char thresh )
```

Threshold of img_input by parameter. if the pixel is below thresh, the output is 0, else, the output is 1. Result is stored in img_output. Input image is 8bpp and output is 1bpp.

2.1.1.40 vglClNdBinToGray()

```
void vglClNdBinToGray (
            VglImage ∗ img_input,
            VglImage ∗ img_output )
```

Convert binary image to grayscale.

Convert binary image to grayscale.

2.2 src/cl2cpp_MM.h File Reference

```
#include "vglImage.h"
#include "vglShape.h"
#include "vglStrEl.h"
```

Functions

- void **vglCl3dFuzzyAlgDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dFuzzyAlgErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dFuzzyArithDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dFuzzyArithErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dFuzzyBoundDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dFuzzyBoundErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dFuzzyDaPDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z, float gama)
- void **vglCl3dFuzzyDaPErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z, float gama)
- void **vglCl3dFuzzyDrasticDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dFuzzyDrasticErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dFuzzyGeoDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dFuzzyGeoErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dFuzzyHamacherDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_↩ window, int window_size_x, int window_size_y, int window_size_z, float gama)
- void **vglCl3dFuzzyHamacherErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_↩ window, int window_size_x, int window_size_y, int window_size_z, float gama)
- void **vglCl3dFuzzyStdDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dFuzzyStdErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglClFuzzyAlgDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)
- void **vglClFuzzyAlgErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)
- void **vglClFuzzyArithDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)
- void **vglClFuzzyArithErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)
- void **vglClFuzzyBoundDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)
- void **vglClFuzzyBoundErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)
- void **vglClFuzzyDaPDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, float gama)
- void **vglClFuzzyDaPErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, float gama)
- void **vglClFuzzyDrasticDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)
- void **vglClFuzzyDrasticErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)
- void **vglClFuzzyGeoDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)

- void **vglClFuzzyGeoErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)
- void **vglClFuzzyHamacherDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, float gama)
- void **vglClFuzzyHamacherErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, float gama)
- void **vglClFuzzyStdDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)
- void **vglClFuzzyStdErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)

2.2.1 Function Documentation

2.2.1.1 vglCl3dFuzzyAlgDilate()

```
void vglCl3dFuzzyAlgDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.2 vglCl3dFuzzyAlgErode()

```
void vglCl3dFuzzyAlgErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.3 vglCl3dFuzzyArithDilate()

```
void vglCl3dFuzzyArithDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.4 vglCl3dFuzzyArithErode()

```
void vglCl3dFuzzyArithErode (
             VglImage * img_input,
             VglImage * img_output,
             float * convolution_window,
             int window_size_x,
             int window_size_y,
             int window_size_z )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.5 vglCl3dFuzzyBoundDilate()

```
void vglCl3dFuzzyBoundDilate (
             VglImage * img_input,
             VglImage * img_output,
             float * convolution_window,
             int window_size_x,
             int window_size_y,
             int window_size_z )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.6 vglCl3dFuzzyBoundErode()

```
void vglCl3dFuzzyBoundErode (
             VglImage * img_input,
             VglImage * img_output,
             float * convolution_window,
             int window_size_x,
             int window_size_y,
             int window_size_z )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.7 vglCl3dFuzzyDaPDilate()

```
void vglCl3dFuzzyDaPDilate (
             VglImage * img_input,
             VglImage * img_output,
             float * convolution_window,
             int window_size_x,
             int window_size_y,
             int window_size_z,
             float gama )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.8 vglCl3dFuzzyDaPErode()

```
void vglCl3dFuzzyDaPErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z,
            float gama )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.9 vglCl3dFuzzyDrasticDilate()

```
void vglCl3dFuzzyDrasticDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.10 vglCl3dFuzzyDrasticErode()

```
void vglCl3dFuzzyDrasticErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.11 vglCl3dFuzzyGeoDilate()

```
void vglCl3dFuzzyGeoDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.12 vglCl3dFuzzyGeoErode()

```
void vglCl3dFuzzyGeoErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.13 vglCl3dFuzzyHamacherDilate()

```
void vglCl3dFuzzyHamacherDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z,
            float gama )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.14 vglCl3dFuzzyHamacherErode()

```
void vglCl3dFuzzyHamacherErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z,
            float gama )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.15 vglCl3dFuzzyStdDilate()

```
void vglCl3dFuzzyStdDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.16 vglCl3dFuzzyStdErode()

```
void vglCl3dFuzzyStdErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.17 vglClFuzzyAlgDilate()

```
void vglClFuzzyAlgDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.18 vglClFuzzyAlgErode()

```
void vglClFuzzyAlgErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.19 vglClFuzzyArithDilate()

```
void vglClFuzzyArithDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.20 vglClFuzzyArithErode()

```
void vglClFuzzyArithErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.21 vglClFuzzyBoundDilate()

```
void vglClFuzzyBoundDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.22 vglClFuzzyBoundErode()

```
void vglClFuzzyBoundErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.23 vglClFuzzyDaPDilate()

```
void vglClFuzzyDaPDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            float gama )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.24 vglClFuzzyDaPErode()

```
void vglClFuzzyDaPErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            float gama )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.25 vglClFuzzyDrasticDilate()

```
void vglClFuzzyDrasticDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.26 vglClFuzzyDrasticErode()

```
void vglClFuzzyDrasticErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.27 vglClFuzzyGeoDilate()

```
void vglClFuzzyGeoDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.28 vglClFuzzyGeoErode()

```
void vglClFuzzyGeoErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.29 vglClFuzzyHamacherDilate()

```
void vglClFuzzyHamacherDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            float gama )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.30 vglClFuzzyHamacherErode()

```
void vglClFuzzyHamacherErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            float gama )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.31 vglClFuzzyStdDilate()

```
void vglClFuzzyStdDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Erosion of src image by mask. Result is stored in dst image.

2.2.1.32 vglClFuzzyStdErode()

```
void vglClFuzzyStdErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Erosion of src image by mask. Result is stored in dst image.

2.3 src/cl2cpp_ND.h File Reference

```
#include "vglImage.h"
#include "vglShape.h"
#include "vglStrEl.h"
```

Functions

- void **vglClNdConvolution** (VglImage ∗img_input, VglImage ∗img_output, VglStrEl ∗window)
- void **vglClNdCopy** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglClNdDilate** (VglImage ∗img_input, VglImage ∗img_output, VglStrEl ∗window)
- void **vglClNdErode** (VglImage ∗img_input, VglImage ∗img_output, VglStrEl ∗window)
- void **vglClNdNot** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglClNdThreshold** (VglImage ∗img_input, VglImage ∗img_output, unsigned char thresh, unsigned char top=255)

2.3.1 Function Documentation

2.3.1.1 vglClNdConvolution()

```
void vglClNdConvolution (
            VglImage * img_input,
            VglImage * img_output,
            VglStrEl * window )
```

N-dimensional convolution

SHAPE directive passes a structure with size of each dimension, offsets and number of dimensions. Parameter does not appear in wrapper parameter list. The C expression between parenthesis returns the desired shape of type VglClShape.

2.3.1.2 vglClNdCopy()

```
void vglClNdCopy (
            VglImage * img_input,
            VglImage * img_output )
```

Copy N-dimensional image.

2.3.1.3 vglClNdDilate()

```
void vglClNdDilate (
            VglImage * img_input,
            VglImage * img_output,
            VglStrEl * window )
```

N-dimensional dilation

SHAPE directive passes a structure with size of each dimension, offsets and number of dimensions. Parameter does not appear in wrapper parameter list. The C expression between parenthesis returns the desired shape of type VglClShape.

2.3.1.4 vglClNdErode()

```
void vglClNdErode (
            VglImage * img_input,
            VglImage * img_output,
            VglStrEl * window )
```

N-dimensional erosion

SHAPE directive passes a structure with size of each dimension, offsets and number of dimensions. Parameter does not appear in wrapper parameter list. The C expression between parenthesis returns the desired shape of type VglClShape.

2.3.1.5 vglClNdNot()

```
void vglClNdNot (
            VglImage * img_input,
            VglImage * img_output )
```

Invert N-dimensional image.

2.3.1.6 vglClNdThreshold()

```
void vglClNdThreshold (
            VglImage * img_input,
            VglImage * img_output,
            unsigned char thresh,
            unsigned char top = 255 )
```

Threshold of img_input by parameter. if the pixel is below thresh, the output is 0, else, the output is top. Result is stored in img_output.

2.4 src/cl2cpp_shaders.h File Reference

```
#include "vglImage.h"
#include "vglShape.h"
#include "vglStrEl.h"
```

Functions

- void **vglCl3dBlurSq3** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglCl3dConvolution** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y, int window_size_z)
- void **vglCl3dCopy** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglCl3dDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window↩ _size_x, int window_size_y, int window_size_z)
- void **vglCl3dErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window↩ _size_x, int window_size_y, int window_size_z)
- void **vglCl3dMax** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglCl3dMin** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglCl3dNot** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglCl3dSub** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglCl3dSum** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglCl3dThreshold** (VglImage ∗src, VglImage ∗dst, float thresh, float top=1.0)
- void **vglClBlurSq3** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglClConvolution** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_size_x, int window_size_y)
- void **vglClCopy** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglClDilate** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_↩ size_x, int window_size_y)
- void **vglClErode** (VglImage ∗img_input, VglImage ∗img_output, float ∗convolution_window, int window_↩ size_x, int window_size_y)
- void **vglClInvert** (VglImage ∗img_input, VglImage ∗img_output)
- void **vglClMax** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglClMin** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglClSub** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglClSum** (VglImage ∗img_input1, VglImage ∗img_input2, VglImage ∗img_output)
- void **vglClSwapRgb** (VglImage ∗src, VglImage ∗dst)
- void **vglClThreshold** (VglImage ∗src, VglImage ∗dst, float thresh, float top=1.0)

2.4.1 Function Documentation

2.4.1.1 vglCl3dBlurSq3()

```
void vglCl3dBlurSq3 (
            VglImage * img_input,
            VglImage * img_output )
```

Convolution of src image by mask. Result is stored in dst image.

In some OpenCL versions, the next directive is required #pragma OPENCL EXTENSION cl_khr_3d_image_writes : enable

2.4.1.2 vglCl3dConvolution()

```
void vglCl3dConvolution (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Convolution of src image by mask. Result is stored in dst image.

2.4.1.3 vglCl3dCopy()

```
void vglCl3dCopy (
            VglImage * img_input,
            VglImage * img_output )
```

Direct copy from src to dst.

2.4.1.4 vglCl3dDilate()

```
void vglCl3dDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Erosion of src image by mask. Result is stored in dst image.

2.4.1.5 vglCl3dErode()

```
void vglCl3dErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y,
            int window_size_z )
```

Erosion of src image by mask. Result is stored in dst image.

2.4.1.6 vglCl3dMax()

```
void vglCl3dMax (
            VglImage * img_input1,
            VglImage * img_input2,
            VglImage * img_output )
```

Direct copy from src to dst.

2.4.1.7 vglCl3dMin()

```
void vglCl3dMin (
            VglImage * img_input1,
            VglImage * img_input2,
            VglImage * img_output )
```

Direct copy from src to dst.

2.4.1.8 vglCl3dNot()

```
void vglCl3dNot (
            VglImage * img_input,
            VglImage * img_output )
```

Direct copy from src to dst.

2.4.1.9 vglCl3dSub()

```
void vglCl3dSub (
            VglImage * img_input1,
            VglImage * img_input2,
            VglImage * img_output )
```

Direct copy from src to dst.

2.4.1.10 vglCl3dSum()

```
void vglCl3dSum (
            VglImage * img_input1,
            VglImage * img_input2,
            VglImage * img_output )
```

Direct copy from src to dst.

2.4.1.11 vglCl3dThreshold()

```
void vglCl3dThreshold (
            VglImage * src,
            VglImage * dst,
            float thresh,
            float top = 1.0 )
```

Threshold of src image by float parameter. if the pixel is below thresh, the output is 0, else, the output is top. Result is stored in dst image.

2.4.1.12 vglClBlurSq3()

```
void vglClBlurSq3 (
            VglImage * img_input,
            VglImage * img_output )
```

Convolution of src image by mask. Result is stored in dst image.

2.4.1.13 vglClConvolution()

```
void vglClConvolution (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Convolution of src image by mask. Result is stored in dst image.

2.4.1.14 vglClCopy()

```
void vglClCopy (
            VglImage * img_input,
            VglImage * img_output )
```

Direct copy from src to dst.

2.4.1.15 vglClDilate()

```
void vglClDilate (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Erosion of src image by mask. Result is stored in dst image.

2.4.1.16 vglClErode()

```
void vglClErode (
            VglImage * img_input,
            VglImage * img_output,
            float * convolution_window,
            int window_size_x,
            int window_size_y )
```

Erosion of src image by mask. Result is stored in dst image.

2.4.1.17 vglClInvert()

```
void vglClInvert (
            VglImage * img_input,
            VglImage * img_output )
```

Negative of src image. Result is stored in dst image.

2.4.1.18 vglClMax()

```
void vglClMax (
            VglImage * img_input1,
            VglImage * img_input2,
            VglImage * img_output )
```

Direct copy from src to dst.

2.4.1.19 vglClMin()

```
void vglClMin (
            VglImage * img_input1,
            VglImage * img_input2,
            VglImage * img_output )
```

Direct copy from src to dst.

2.4.1.20 vglClSub()

```
void vglClSub (
            VglImage * img_input1,
            VglImage * img_input2,
            VglImage * img_output )
```

Direct copy from src to dst.

2.4.1.21 vglClSum()

```
void vglClSum (
            VglImage * img_input1,
            VglImage * img_input2,
            VglImage * img_output )
```

Direct copy from src to dst.

2.4.1.22 vglClSwapRgb()

```
void vglClSwapRgb (
            VglImage * src,
            VglImage * dst )
```

Swap R and B channels.

2.4.1.23 vglClThreshold()

```
void vglClThreshold (
            VglImage * src,
            VglImage * dst,
            float thresh,
            float top = 1.0 )
```

Threshold of src image by float parameter. if the pixel is below thresh, the output is 0, else, the output is top. Result is stored in dst image.

2.5 src/glsl2cpp_BG.h File Reference

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

Functions

- void **vglDetectFGSimpleBGModel** (VglImage ∗img_in, VglImage ∗average, VglImage ∗variance, VglImage ∗foreground, float std_thresh)
- void **vglTrainSimpleBGModel** (VglImage ∗img_in, VglImage ∗average, VglImage ∗variance, float weight)
- void **vglUpdatePartialSimpleBGModel** (VglImage ∗img_in, VglImage ∗foregorundClose, VglImage ∗average, VglImage ∗variance, float weight)

2.5.1 Function Documentation

2.5.1.1 vglDetectFGSimpleBGModel()

```
void vglDetectFGSimpleBGModel (
            VglImage * img_in,
            VglImage * average,
            VglImage * variance,
            VglImage * foreground,
            float std_thresh )
```

Detects foreground pixels.

2.5.1.2 vglTrainSimpleBGModel()

```
void vglTrainSimpleBGModel (
            VglImage * img_in,
            VglImage * average,
            VglImage * variance,
            float weight )
```

Updates average and variance of background model.

2.5.1.3 vglUpdatePartialSimpleBGModel()

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

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

2.6 src/glsl2cpp_shaders.h File Reference

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

Functions

- void **shader_15_1** (VglImage ∗src, VglImage ∗dst)
- void **vgl1to3Channels** (VglImage ∗src, VglImage ∗dst)
- void **vgl3dNot** (VglImage ∗src, VglImage ∗dst)
- void **vglAbsDiff** (VglImage ∗src0, VglImage ∗src1, VglImage ∗dst)
- void **vglAnd** (VglImage ∗src0, VglImage ∗src1, VglImage ∗dst)
- void **vglBaricenterInit** (VglImage ∗src, VglImage ∗dst)
- void **vglBlurSq3** (VglImage ∗src, VglImage ∗dst)
- void **vglClear2** (VglImage ∗src_dst, float r, float g, float b, float a=0.0)
- void **vglContrast** (VglImage ∗src, VglImage ∗dst, float factor)
- void **vglCoordToColor** (VglImage ∗dst)
- void **vglCopy** (VglImage ∗src, VglImage ∗dst)
- void **vglCrossingNumber** (VglImage ∗src, VglImage ∗dst)
- void **vglDeleteSkeletonCorners** (VglImage ∗src, VglImage ∗dst, int step)
- void **vglDeleteSkeletonWarts** (VglImage ∗src, VglImage ∗dst)
- void **vglDeleteSkeletonWarts2** (VglImage ∗src, VglImage ∗dst)
- void **vglDiff** (VglImage ∗src0, VglImage ∗src1, VglImage ∗dst)
- void **vglDilateCross3** (VglImage ∗src, VglImage ∗dst)
- void **vglDilateSq3** (VglImage ∗src, VglImage ∗dst)
- void **vglErodeCross3** (VglImage ∗src, VglImage ∗dst)
- void **vglErodeHL3** (VglImage ∗src, VglImage ∗dst)
- void **vglErodeHL5** (VglImage ∗src, VglImage ∗dst)
- void **vglErodeHL7** (VglImage ∗src, VglImage ∗dst)
- void **vglErodeSq3** (VglImage ∗src, VglImage ∗dst)
- void **vglErodeSq3off** (VglImage ∗src, VglImage ∗dst)
- void **vglErodeSq5** (VglImage ∗src, VglImage ∗dst)
- void **vglErodeSq5off** (VglImage ∗src, VglImage ∗dst)
- void **vglErodeSq7** (VglImage ∗src, VglImage ∗dst)
- void **vglErodeSqSide** (VglImage ∗src, VglImage ∗dst, int side)
- void **vglErodeVL3** (VglImage ∗src, VglImage ∗dst)
- void **vglErodeVL5** (VglImage ∗src, VglImage ∗dst)
- void **vglErodeVL7** (VglImage ∗src, VglImage ∗dst)
- void **vglFeaturePoints** (VglImage ∗src, VglImage ∗dst, int type)
- void **vglGaussianBlurSq3** (VglImage ∗src, VglImage ∗dst)
- void **vglGray** (VglImage ∗src, VglImage ∗dst)
- void **vglHorizontalFlip** (VglImage ∗src, VglImage ∗dst)
- void **vglInOut** (VglImage ∗src, VglImage ∗dst)
- void **vglJulia** (VglImage ∗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** (VglImage ∗src, VglImage ∗dst)
- void **vglMandel** (VglImage ∗dst, float ox=0.0, float oy=0.0, float half_win=1.0)
- void **vglMipmap** (VglImage ∗src, VglImage ∗dst, float lod)
- void **vglMulScalar** (VglImage ∗src, VglImage ∗dst, float factor)
- void **vglMultiInput** (VglImage ∗src0, VglImage ∗src1, VglImage ∗dst, float weight=.5)
- void **vglMultiOutput** (VglImage ∗src, VglImage ∗dst, VglImage ∗dst1)
- void **vglNoise** (VglImage ∗src, VglImage ∗dst)
- void **vglNot** (VglImage ∗src, VglImage ∗dst)
- void **vglOr** (VglImage ∗src0, VglImage ∗src1, VglImage ∗dst)
- void **vglRescale** (VglImage ∗src, VglImage ∗dst, float x0, float y0, float x1, float y1)
- void **vglRgbToBgr** (VglImage ∗src, VglImage ∗dst)
- void **vglRgbToHsl** (VglImage ∗src, VglImage ∗dst)
- void **vglRgbToHsv** (VglImage ∗src, VglImage ∗dst)
- void **vglRgbToXyz** (VglImage ∗src, VglImage ∗dst)
- void **vglRobertsGradient** (VglImage ∗src, VglImage ∗dst)

- void **vglSelfSum22** (VglImage ∗src, VglImage ∗dst)
- void **vglSelfSum3v** (VglImage ∗src, VglImage ∗dst)
- void **vglSelfSum4h** (VglImage ∗src, VglImage ∗dst)
- void **vglSelfSum5h** (VglImage ∗src, VglImage ∗dst)
- void **vglSelfSum5v** (VglImage ∗src, VglImage ∗dst)
- void **vglSharpenSq3** (VglImage ∗src, VglImage ∗dst)
- void **vglSobelGradient** (VglImage ∗src, VglImage ∗dst)
- void **vglSobelXSq3** (VglImage ∗src, VglImage ∗dst)
- void **vglSobelYSq3** (VglImage ∗src, VglImage ∗dst)
- void **vglSum** (VglImage ∗src0, VglImage ∗src1, VglImage ∗dst)
- void **vglSumWeighted** (VglImage ∗src0, VglImage ∗src1, VglImage ∗dst, float weight=.5)
- void **vglSwapRGB** (VglImage ∗src, VglImage ∗dst)
- void **vglTestInOut** (VglImage ∗src_dst, float r, float g, float b, float a=0.0)
- void **vglTestInOut2** (VglImage ∗src_dst, VglImage ∗dst)
- void **vglTestMultiInput** (VglImage ∗src0, VglImage ∗src1, VglImage ∗dst, float weight=.5)
- void **vglTestMultiOutput** (VglImage ∗src, VglImage ∗dst, VglImage ∗dst1)
- void **vglTeste** (VglImage ∗src, VglImage ∗dst)
- void **vglThinBernardAux** (VglImage ∗src, VglImage ∗eroded, VglImage ∗dst)
- void **vglThinChinAux** (VglImage ∗src, VglImage ∗dst)
- void **vglThresh** (VglImage ∗src, VglImage ∗dst, float thresh, float top=1.0)
- void **vglThreshLevelSet** (VglImage ∗src, VglImage ∗dst, float thresh, float top=1.0)
- void **vglVerticalFlip** (VglImage ∗src, VglImage ∗dst)
- void **vglWhiteRohrerEdge** (VglImage ∗src, VglImage ∗dst, float radius)
- void **vglXGY** (VglImage ∗src, VglImage ∗dst)
- void **vglZoom** (VglImage ∗src, VglImage ∗dst, float factor)

2.6.1 Function Documentation

2.6.1.1 shader_15_1()

```
void shader_15_1 (
            VglImage * src,
            VglImage * dst )
```

2.6.1.2 vgl1to3Channels()

```
void vgl1to3Channels (
            VglImage * src,
            VglImage * dst )
```

Convert grayscale image to RGB

2.6.1.3 vgl3dNot()

```
void vgl3dNot (
            VglImage * src,
            VglImage * dst )
```

Inverts 3d image.

As the wrappers are implemented currently, the shader will invert only the first layer of the 3d image.

2.6.1.4 vglAbsDiff()

```
void vglAbsDiff (
            VglImage * src0,
            VglImage * src1,
            VglImage * dst )
```

Absolute difference between two images.

Referenced by vglGetLevelDistTransform5().

2.6.1.5 vglAnd()

```
void vglAnd (
            VglImage * src0,
            VglImage * src1,
            VglImage * dst )
```

Logical AND between two images

2.6.1.6 vglBaricenterInit()

```
void vglBaricenterInit (
            VglImage * src,
            VglImage * 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 th 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().

2.6.1.7 vglBlurSq3()

```
void vglBlurSq3 (
            VglImage * src,
            VglImage * dst )
```

vglBlurSq3

Blur image by 3x3 square structuring element.

2.6.1.8 vglClear2()

```
void vglClear2 (
            VglImage * src_dst,
            float r,
            float g,
            float b,
            float a = 0.0 )
```

Clear image with given color.

2.6.1.9 vglContrast()

```
void vglContrast (
            VglImage * src,
            VglImage * dst,
            float factor )
```

Changes contrast of image by given factor.

2.6.1.10 vglCoordToColor()

```
void vglCoordToColor (
            VglImage * 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.

2.6.1.11 vglCopy()

```
void vglCopy (
            VglImage * src,
            VglImage * dst )
```

Direct copy from src to dst.

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

2.6.1.12 vglCrossingNumber()

```
void vglCrossingNumber (
            VglImage * src,
            VglImage * dst )
```

Crossing number is defined as the number of ocurrences of the pattern 01 in the neihborhood of a pixel.

```
Neighborhood of pixel P is indexed as follows:
```

$$\begin{array}{ccc} P3 & P2 & P1 \\ P4 & P & P0/8 \\ P5 & P6 & P7 \end{array}$$

```
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
```

2.6.1.13 vglDeleteSkeletonCorners()

```
void vglDeleteSkeletonCorners (
            VglImage * src,
            VglImage * 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:

$$\begin{array}{ccc} P3 & P2 & P1 \\ P4 & P8 & P0 \\ P5 & P6 & P7 \end{array}$$

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

$$\begin{array}{ccc} 0 & 0 & x \\ 0 & 1 & 1 \\ x & 1 & 0 \end{array}$$

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

2.6.1.14 vglDeleteSkeletonWarts()

```
void vglDeleteSkeletonWarts (
             VglImage * src,
             VglImage * dst )
```

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

$$\begin{array}{ccc} P3 & P2 & P1 \\ P4 & P & P0/8 \\ P5 & P6 & P7 \end{array}$$

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

$$\begin{array}{ccc} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \end{array}$$

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
```

2.6.1.15 vglDeleteSkeletonWarts2()

```
void vglDeleteSkeletonWarts2 (
             VglImage * src,
             VglImage * 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 >=3

```
References:

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

2.6.1.16 vglDiff()

```
void vglDiff (
             VglImage * src0,
             VglImage * src1,
             VglImage * dst )
```

Image src0 minus src1.

2.6.1.17 vglDilateCross3()

```
void vglDilateCross3 (
             VglImage * src,
             VglImage * dst )
```

Dilation of image by 3x3 cross structuring element.

2.6.1.18 vglDilateSq3()

```
void vglDilateSq3 (
             VglImage * src,
             VglImage * dst )
```

Dilation of image by 3x3 square structuring element.

Referenced by vglCloseSq3(), and vglOpenSq3().

2.6.1.19 vglErodeCross3()

```
void vglErodeCross3 (
             VglImage * src,
             VglImage * dst )
```

Erosion of image by 3x3 cross structuring element.

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

2.6.1.20 vglErodeHL3()

```
void vglErodeHL3 (
             VglImage * src,
             VglImage * dst )
```

Erosion of image by horizontal line with 3 pixels.

Referenced by vglErodeSq3Sep().

2.6.1.21 vglErodeHL5()

```
void vglErodeHL5 (
            VglImage * src,
            VglImage * dst )
```

Erosion of image by horizontal line with 5 pixels.

Referenced by vglErodeSq5Sep().

2.6.1.22 vglErodeHL7()

```
void vglErodeHL7 (
            VglImage * src,
            VglImage * dst )
```

Erosion of image by horizontal line with 7 pixels.

2.6.1.23 vglErodeSq3()

```
void vglErodeSq3 (
            VglImage * src,
            VglImage * dst )
```

Erosion of image by 3x3 square structuring element.

Referenced by vglCloseSq3(), vglDistTransform5(), vglDistTransformSq3(), vglGetLevelDistTransform5(), and vgl↵ OpenSq3().

2.6.1.24 vglErodeSq3off()

```
void vglErodeSq3off (
            VglImage * src,
            VglImage * dst )
```

Erosion of image by 3x3 square structuring element. Uses an offset array with 9 elements. Slower than vglErode↵ Sq3.

2.6.1.25 vglErodeSq5()

```
void vglErodeSq5 (
            VglImage * src,
            VglImage * dst )
```

Erosion of image by 5x5 square structuring element.

2.6.1.26 vglErodeSq5off()

```
void vglErodeSq5off (
            VglImage * src,
            VglImage * dst )
```

Erosion of image by 3x3 square structuring element. Uses an offset array with 25 elements. Slower than vgl↩ ErodeSq5.

2.6.1.27 vglErodeSq7()

```
void vglErodeSq7 (
            VglImage * src,
            VglImage * dst )
```

Erosion of image by 7x7 square structuring element.

2.6.1.28 vglErodeSqSide()

```
void vglErodeSqSide (
            VglImage * src,
            VglImage * dst,
            int side )
```

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

2.6.1.29 vglErodeVL3()

```
void vglErodeVL3 (
            VglImage * src,
            VglImage * dst )
```

Erosion of image by vertical line with 3 pixels.

Referenced by vglErodeSq3Sep().

2.6.1.30 vglErodeVL5()

```
void vglErodeVL5 (
            VglImage * src,
            VglImage * dst )
```

Erosion of image by vertical line with 5 pixels.

Referenced by vglErodeSq5Sep().

2.6.1.31 vglErodeVL7()

```
void vglErodeVL7 (
             VglImage * src,
             VglImage * dst )
```

Erosion of image by vertical line with 7 pixels.

2.6.1.32 vglFeaturePoints()

```
void vglFeaturePoints (
             VglImage * src,
             VglImage * 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 | Nc(P) = 1 }

```
Feature points type 1, denoted as S1, are defined as
```

S1 = { P | Nc(P) >= 3}

```
Feature points type 2, denoted as S2, are defined as
```

S1 = { P | Nb(P) >= 3}

```
Feature points type 3, denoted as S3, are defined as
```

S3 = { P | Nc(P) >= 3 or Nb(P) >= 4 }

```
References:

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

2.6.1.33 vglGaussianBlurSq3()

```
void vglGaussianBlurSq3 (
            VglImage * src,
            VglImage * dst )
```

Blurs image by 3x3 square gaussian structuring element.

2.6.1.34 vglGray()

```
void vglGray (
            VglImage * src,
            VglImage * dst )
```

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

2.6.1.35 vglHorizontalFlip()

```
void vglHorizontalFlip (
            VglImage * src,
            VglImage * dst )
```

Flip image horizontally i.e. left becomes right.

Image flip done by shader.

2.6.1.36 vglInOut()

```
void vglInOut (
            VglImage * src,
            VglImage * dst )
```

vglInOut

Test and model for IN_OUT semantics

2.6.1.37 vglJulia()

```
void vglJulia (
            VglImage * 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

2.6.1.38 vglLaplaceSq3()

```
void vglLaplaceSq3 (
            VglImage * src,
            VglImage * dst )
```

Laplacian of image by 3x3 square structuring element.

2.6.1.39 vglMandel()

```
void vglMandel (
            VglImage * dst,
            float ox = 0.0,
            float oy = 0.0,
            float half_win = 1.0 )
```

Calculate Mandelbrot set

2.6.1.40 vglMipmap()

```
void vglMipmap (
            VglImage * src,
            VglImage * dst,
            float lod )
```

Get specified level of detail.

2.6.1.41 vglMulScalar()

```
void vglMulScalar (
            VglImage * src,
            VglImage * dst,
            float factor )
```

Multiply image by scalar.

2.6.1.42 vglMultiInput()

```
void vglMultiInput (
            VglImage * src0,
            VglImage * src1,
            VglImage * dst,
            float weight = .5 )
```

VglAdd

Sum of two images.

2.6.1.43 vglMultiOutput()

```
void vglMultiOutput (
            VglImage * src,
            VglImage * dst,
            VglImage * dst1 )
```

vglGray

Convert image to grayscale

2.6.1.44 vglNoise()

```
void vglNoise (
            VglImage * src,
            VglImage * dst )
```

Add gaussian noise to image

2.6.1.45 vglNot()

```
void vglNot (
            VglImage * src,
            VglImage * dst )
```

Inverts image.

2.6.1.46 vglOr()

```
void vglOr (
            VglImage * src0,
            VglImage * src1,
            VglImage * dst )
```

Logical OR between two images

Referenced by vglCErodeCross3().

2.6.1.47 vglRescale()

```
void vglRescale (
            VglImage * src,
            VglImage * dst,
            float x0,
            float y0,
            float x1,
            float y1 )
```

Rescales corners of image to given corners

2.6.1.48 vglRgbToBgr()

```
void vglRgbToBgr (
            VglImage * src,
            VglImage * dst )
```

Converts image RGB to BGR color space

2.6.1.49 vglRgbToHsl()

```
void vglRgbToHsl (
            VglImage * src,
            VglImage * dst )
```

Converts image RGB to HSL color space

2.6.1.50 vglRgbToHsv()

```
void vglRgbToHsv (
            VglImage * src,
            VglImage * dst )
```

Converts image RGB to HSV color space

2.6.1.51 vglRgbToXyz()

```
void vglRgbToXyz (
            VglImage * src,
            VglImage * dst )
```

Converts image RGB to XYZ color space.

2.6.1.52 vglRobertsGradient()

```
void vglRobertsGradient (
            VglImage * src,
            VglImage * dst )
```

Roberts gradient of image

2.6.1.53 vglSelfSum22()

```
void vglSelfSum22 (
            VglImage * src,
            VglImage * 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().

2.6.1.54 vglSelfSum3v()

```
void vglSelfSum3v (
            VglImage * src,
            VglImage * 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().

2.6.1.55 vglSelfSum4h()

```
void vglSelfSum4h (
            VglImage * src,
            VglImage * 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().

2.6.1.56 vglSelfSum5h()

```
void vglSelfSum5h (
            VglImage * src,
            VglImage * 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().

2.6.1.57 vglSelfSum5v()

```
void vglSelfSum5v (
            VglImage * src,
            VglImage * 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().

2.6.1.58 vglSharpenSq3()

```
void vglSharpenSq3 (
             VglImage * src,
             VglImage * dst )
```

Sharpens image using 3x3 square window.

2.6.1.59 vglSobelGradient()

```
void vglSobelGradient (
             VglImage * src,
             VglImage * dst )
```

Sobel gradient of image

2.6.1.60 vglSobelXSq3()

```
void vglSobelXSq3 (
             VglImage * src,
             VglImage * dst )
```

Sobel edge filtering in X direction.

2.6.1.61 vglSobelYSq3()

```
void vglSobelYSq3 (
             VglImage * src,
             VglImage * dst )
```

Sobel edge filtering in Y direction.

2.6.1.62 vglSum()

```
void vglSum (
             VglImage * src0,
             VglImage * src1,
             VglImage * dst )
```

Sum of two images.

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

2.6.1.63 vglSumWeighted()

```
void vglSumWeighted (
            VglImage * src0,
            VglImage * src1,
            VglImage * 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.

2.6.1.64 vglSwapRGB()

```
void vglSwapRGB (
            VglImage * src,
            VglImage * dst )
```

Convert image from RGB to BGR and vice versa.

2.6.1.65 vglTeste()

```
void vglTeste (
            VglImage * src,
            VglImage * dst )
```

vglDilate

Dilation of image by 3x3 square structuring element.

2.6.1.66 vglTestInOut()

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

Test and model for IN_OUT semantics

2.6.1.67 vglTestInOut2()

```
void vglTestInOut2 (
            VglImage * src_dst,
            VglImage * dst )
```

Test and model for IN_OUT semantics, with double output.

2.6.1.68 vglTestMultiInput()

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

Test and model for multiple input functions.

2.6.1.69 vglTestMultiOutput()

```
void vglTestMultiOutput (
            VglImage * src,
            VglImage * dst,
            VglImage * dst1 )
```

Test and model for multiple output functions.

2.6.1.70 vglThinBernardAux()

```
void vglThinBernardAux (
            VglImage * src,
            VglImage * eroded,
            VglImage * 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:

$$\begin{array}{ccc} P3 & P2 & P1 \\ P4 & P8 & P0 \\ P5 & P6 & P7 \end{array}$$

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().

2.6.1.71 vglThinChinAux()

```
void vglThinChinAux (
            VglImage * src,
            VglImage * 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:

$$\begin{array}{ccccc} x & x & P10 & x & x \\ x & P3 & P2 & P1 & x \\ P11 & P4 & P0 & P8 & P9 \\ x & P5 & P6 & P7 & x \\ x & x & P12 & x & x \end{array}$$

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().

2.6.1.72 vglThresh()

```
void vglThresh (
            VglImage * src,
            VglImage * 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().

2.6.1.73 vglThreshLevelSet()

```
void vglThreshLevelSet (
            VglImage * src,
            VglImage * 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.

2.6.1.74 vglVerticalFlip()

```
void vglVerticalFlip (
            VglImage * src,
            VglImage * dst )
```

Flip image vertically i.e. top becomes bottom.

Image flip done by shader.

2.6.1.75 vglWhiteRohrerEdge()

```
void vglWhiteRohrerEdge (
            VglImage * src,
            VglImage * dst,
            float radius )
```

Finds edge by using a White-Rohrer mask.

2.6.1.76 vglXGY()

```
void vglXGY (
            VglImage * src,
            VglImage * dst )
```

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

2.6.1.77 vglZoom()

```
void vglZoom (
            VglImage * src,
            VglImage * dst,
            float factor )
```

Zoom image by factor.

2.7 src/glsl2cpp_Stereo.h File Reference

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

Functions

- void **vglAbsDiffDisparity** (VglImage ∗img_ref, VglImage ∗img_2, VglImage ∗dst, float disparity)
- void **vglAbsDiffDisparityMipmap** (VglImage ∗img_ref, VglImage ∗img_2, VglImage ∗dst, float disparity, float max_lod)
- void **vglFindDisparity** (VglImage ∗img_dif, VglImage ∗img_disp, float disparity)
- void **vglFindDisparityDiff** (VglImage ∗img_sum, VglImage ∗img_disp, VglImage ∗img_best, float disparity)
- void **vglGreenDiffDisparity** (VglImage ∗img_ref, VglImage ∗img_2, VglImage ∗dst, float disparity)
- void **vglHomography** (VglImage ∗img_src, VglImage ∗img_dst, float ∗f_homo)
- void **vglMapTo3D** (VglImage ∗img_map, VglImage ∗img_3d, float f, float b, float D, float disp_k=0.0, float h=10.0)
- void **vglMeanMipmap** (VglImage ∗img_dif, VglImage ∗img_out, float max_lod)
- void **vglMeanSq3** (VglImage ∗img_dif, VglImage ∗img_out)
- void **vglRectify** (VglImage ∗img_src, VglImage ∗img_dst, float ∗f_dist, float ∗f_proj, float ∗f_homo)
- void **vglSumDiff** (VglImage ∗img_dif, VglImage ∗img_out)
- void **vglSumDiffMipmap** (VglImage ∗img_dif, VglImage ∗img_out, float max_lod)
- void **vglUndistort** (VglImage ∗img_src, VglImage ∗img_dst, float ∗f_dist, float ∗f_proj)

2.7.1 Function Documentation

2.7.1.1 vglAbsDiffDisparity()

```
void vglAbsDiffDisparity (
            VglImage * img_ref,
            VglImage * img_2,
            VglImage * 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.

2.7.1.2 vglAbsDiffDisparityMipmap()

```
void vglAbsDiffDisparityMipmap (
            VglImage * img_ref,
            VglImage * img_2,
            VglImage * 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.

2.7.1.3 vglFindDisparity()

```
void vglFindDisparity (
            VglImage * img_dif,
            VglImage * 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.

2.7.1.4 vglFindDisparityDiff()

```
void vglFindDisparityDiff (
            VglImage * img_sum,
            VglImage * img_disp,
            VglImage * 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.

2.7.1.5 vglGreenDiffDisparity()

```
void vglGreenDiffDisparity (
            VglImage * img_ref,
            VglImage * img_2,
            VglImage * 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.

2.7.1.6 vglHomography()

```
void vglHomography (
            VglImage * img_src,
            VglImage * img_dst,
            float * f_homo )
```

Apply homography in img_src and stores result in img_dst.

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

`mat2 m` = `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 tbe matrix before using it in OpenGL context.

2.7.1.7 vglMapTo3D()

```
void vglMapTo3D (
            VglImage * img_map,
            VglImage * 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 $*$ (depth / D). if depth == D then z = 0.

disp_k: If set, single disparity will be used.

h: height of camera in cm

2.7.1.8 vglMeanMipmap()

```
void vglMeanMipmap (
            VglImage * img_dif,
            VglImage * img_out,
            float max_lod )
```

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

2.7.1.9 vglMeanSq3()

```
void vglMeanSq3 (
            VglImage * img_dif,
            VglImage * img_out )
```

Mean filter with a 3x3 square mask.

2.7.1.10 vglRectify()

```
void vglRectify (
            VglImage * img_src,
            VglImage * 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 intrinsinc 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 cmponents are written in column major order:

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

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

2.7.1.11 vglSumDiff()

```
void vglSumDiff (
            VglImage * img_dif,
            VglImage * img_out )
```

VglSumDiff

Sum of differences

2.7.1.12 vglSumDiffMipmap()

```
void vglSumDiffMipmap (
            VglImage * img_dif,
            VglImage * img_out,
            float max_lod )
```

VglSumDiffMipmap

Sum of differences

2.7.1.13 vglUndistort()

```
void vglUndistort (
            VglImage * img_src,
            VglImage * 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 intrinsinc 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`

2.8 src/vglImage.cpp File Reference

```
#include <iostream>
#include <stdint.h>
#include <string.h>
#include <malloc.h>
#include <GL/glew.h>
#include <GL/freeglut.h>
#include <vglOpencv.h>
#include "vglContext.h"
#include "vglImage.h"
#include "vglLoadShader.h"
#include "iplImage.h"
#include "glsl2cpp_shaders.h"
```

Functions

- int **vglInit** ()
- int **vglInit** (int w, int h)
- void **vglUpload** (VglImage *image, int swapRGB)
- VglImage * **vglCopyCreateImage** (VglImage *img_in)
- VglImage * **vglCopyCreateImage** (IplImage *img_in, int ndim, int has_mipmap)
- VglImage * **vglCreateImage** (VglImage *img_in)
- VglImage * **vglCreateImage** (IplImage *img_in, int ndim, int has_mipmap)
- VglImage * **vglCreateImage** (int *shape, int depth, int ndim, int has_mipmap)
- VglImage * **vglCreateImage** (VglShape *vglShape, int depth, int has_mipmap)
- VglImage * **vglCreateImage** (CvSize size, int depth, int nChannels, int ndim, int has_mipmap)
- VglImage * **vglCreate3dImage** (CvSize size, int depth, int nChannels, int layers, int has_mipmap)
- VglImage * **vglCreateNdImage** (int ndim, int *shape, int depth, int has_mipmap)
- void **vglSaveImage** (char *filename, VglImage *image)
- void **vglSaveIplImage** (char *filename, IplImage *ipl, int *params)
- void **vglSave3dImage** (char *filename, VglImage *image, int lStart, int lEnd)
- void **vglSaveNdImage** (char *filename, VglImage *image, int lStart, int lEndParam)
- void **vglNdarray3To4Channels** (VglImage *img)

- void **vglNdarray4To3Channels** (VglImage ∗img)
- void **vglIpl3To4Channels** (VglImage ∗img)
- void **vglIpl4To3Channels** (VglImage ∗img)
- void **vglImage3To4Channels** (VglImage ∗img)
- void **vglImage4To3Channels** (VglImage ∗img)
- void **vglReleaseImage** (VglImage ∗∗p_image)
- void **vglReplaceIpl** (VglImage ∗image, IplImage ∗new_ipl)
- void **vglDownloadFaster** (VglImage ∗image)
- void **vglDownload** (VglImage ∗image)
- void **vglDownloadFBO** (VglImage ∗image)
- void **vglDownloadPPM** (VglImage ∗image)
- void **vglDownloadPGM** (VglImage ∗image)
- VglImage ∗ **vglLoadImage** (char ∗filename, int iscolor, int has_mipmap)
- VglImage ∗ **vglLoad3dImage** (char ∗filename, int lStart, int lEnd, bool has_mipmap)
- VglImage ∗ **vglLoadNdImage** (char ∗filename, int lStart, int lEnd, int ∗shape, int ndim, bool has_mipmap)
- int **vglReshape** (VglImage ∗img, VglShape ∗newShape)
- void **iplPrintImageInfo** (IplImage ∗ipl, char ∗msg)
- void **vglPrintImageInfo** (VglImage ∗image, char ∗msg)
- void **vglPrintImageData** (VglImage ∗image, char ∗msg, char ∗format)
- void **iplPrintImageData** (IplImage ∗image, char ∗msg, char ∗format)
- void **vglCopyImageTex** (VglImage ∗src, VglImage ∗dst)
- void **vglCopyImageTexFS** (VglImage ∗src, VglImage ∗dst)
- void **vglCopyImageTexVFS** (VglImage ∗src, VglImage ∗dst)
- void **vglVerticalFlip2** (VglImage ∗src, VglImage ∗dst)
- void **vglHorizontalFlip2** (VglImage ∗src, VglImage ∗dst)
- void **vglClear** (VglImage ∗image, float r, float g, float b, float a)
- void **vglOpenSq3** (VglImage ∗src, VglImage ∗dst, VglImage ∗buf, int times)
- void **vglCloseSq3** (VglImage ∗src, VglImage ∗dst, VglImage ∗buf, int times)
- void **vglErodeSq3Sep** (VglImage ∗src, VglImage ∗dst, VglImage ∗buf, int times)
- void **vglErodeSq5Sep** (VglImage ∗src, VglImage ∗dst, VglImage ∗buf, int times)
- void **vglCErodeCross3** (VglImage ∗src, VglImage ∗mask, VglImage ∗dst, VglImage ∗buf, int times)
- int **SavePPM** (char ∗filename, int w, int h, void ∗savebuf)
- int **vglSavePPM** (char ∗filename, VglImage ∗img)
- int **vglSavePgm** (char ∗filename, VglImage ∗img)
- VglImage ∗ **vglLoadPgm** (char ∗filename)
- int **vglHasDisplay** ()
- int **SaveYUV411** (char ∗filename, int w, int h, void ∗savebuf)
- void **vglDistTransformCross3** (VglImage ∗src, VglImage ∗dst, VglImage ∗buf, VglImage ∗buf2, int times)
- void **vglDistTransformSq3** (VglImage ∗src, VglImage ∗dst, VglImage ∗buf, VglImage ∗buf2, int times)
- void **vglDistTransform5** (VglImage ∗src, VglImage ∗dst, VglImage ∗buf, VglImage ∗buf2, int times)
- void **vglGetLevelDistTransform5** (VglImage ∗src, VglImage ∗dst, VglImage ∗buf, VglImage ∗buf2, int times)
- void **vglThinBernard** (VglImage ∗src, VglImage ∗dst, VglImage ∗buf, int times)
- void **vglThinChin** (VglImage ∗src, VglImage ∗dst, VglImage ∗buf, int times)
- void **vglBaricenterVga** (VglImage ∗src, double ∗x_avg, double ∗y_avg, double ∗pix_count)
- void **vglClForceAsBuf** (VglImage ∗img)
- void **vglMultiOutput_model** (VglImage ∗src, VglImage ∗dst, VglImage ∗dst1)
- void **vglInOut_model** (VglImage ∗dst, VglImage ∗dst1)
- void **vglMultiInput_model** (VglImage ∗src0, VglImage ∗src1, VglImage ∗dst)

2.8.1 Function Documentation

2.8.1.1 iplPrintImageData()

```
void iplPrintImageData (
             IplImage * image,
             char * msg,
             char * format )
```

Print image pixels in text format to stdout

2.8.1.2 iplPrintImageInfo()

```
void iplPrintImageInfo (
             IplImage * ipl,
             char * msg )
```

Print information about image.

Print width, height, depth and number of channels

2.8.1.3 SavePPM()

```
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().

2.8.1.4 SaveYUV411()

```
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.

2.8.1.5 vglBaricenterVga()

```
void vglBaricenterVga (
            VglImage * src,
            double * x_avg,
            double * y_avg,
            double * pix_count )
```

Calculates baricenter of vga image.

The resuld 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().

2.8.1.6 vglCErodeCross3()

```
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().

2.8.1.7 vglClear()

```
void vglClear (
            VglImage * image,
            float r,
            float g,
            float b,
            float a )
```

References vglPrintImageInfo().

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

2.8.1.8 vglClForceAsBuf()

```
void vglClForceAsBuf (
             VglImage * img )
```

Force image to be traated as buffer

Data with 2 and 3 dimensions are, by default, treated as images, i.e. are created and transferred by the API functions clCreate2DImage, clCreate3DImage, clEnqueueWriteImage, clEnqueueReadImage.

To treat data as images has as advantage the possibility of automatic clamping to edge in window operations.

Data with 1, 4 or more dimensions are always treated as buffers, i.e. are created and transferred by the API functions clCreateBuffer, clEnqueueWriteBuffer, clEnqueueReadBuffer.

Use this function to force 2 and 3 dimensions data to be treated as buffers.

Referenced by vglCopyCreateImage(), and vglCreateImage().

2.8.1.9 vglCloseSq3()

```
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().

2.8.1.10 vglCopyCreateImage() [1/2]

```
VglImage* vglCopyCreateImage (
             VglImage * img_in )
```

Create image with same format and data as img_in.

TODO: fix. Not working as expected.

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

2.8.1.11 vglCopyCreateImage() [2/2]

```
VglImage* vglCopyCreateImage (
            IplImage * img_in,
            int ndim,
            int has_mipmap )
```

Create image with same format and data as img_in

References vglCreateImage().

2.8.1.12 vglCopyImageTex()

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

Copy data from src texture to dst texture

2.8.1.13 vglCopyImageTexFS()

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

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

2.8.1.14 vglCopyImageTexVFS()

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

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

2.8.1.15 vglCreate3dImage()

```
VglImage* vglCreate3dImage (
            CvSize size,
            int depth,
            int nChannels,
            int layers,
            int has_mipmap )
```

Create image as described by the parameters

References vglCreateImage().

2.8.1.16 vglCreateImage() [1/5]

```
VglImage* vglCreateImage (
            VglImage * img_in )
```

Create image with same format as img_in

References vglClForceAsBuf().

Referenced by vglBaricenterVga(), vglCopyCreateImage(), vglCreate3dImage(), vglCreateImage(), vglCreateNd←Image(), vglLoad3dImage(), vglLoadImage(), vglLoadNdImage(), and vglLoadPgm().

2.8.1.17 vglCreateImage() [2/5]

```
VglImage* vglCreateImage (
            IplImage * img_in,
            int ndim,
            int has_mipmap )
```

Create image with same format as img_in

References vglCreateImage().

2.8.1.18 vglCreateImage() [3/5]

```
VglImage* vglCreateImage (
            int * shape,
            int depth,
            int ndim,
            int has_mipmap )
```

Create image as described by the parameters

2.8.1.19 vglCreateImage() [4/5]

```
VglImage* vglCreateImage (
            VglShape * vglShape,
            int depth,
            int has_mipmap )
```

Create image as described by the parameters

References vglCreateImage().

2.8.1.20 vglCreateImage() [5/5]

```
VglImage* vglCreateImage (
             CvSize size,
             int depth,
             int nChannels,
             int ndim,
             int has_mipmap )
```

Create image as described by the parameters

References vglCreateImage().

2.8.1.21 vglCreateNdImage()

```
VglImage* vglCreateNdImage (
             int ndim,
             int * shape,
             int depth,
             int has_mipmap )
```

Create image as described by the parameters

References vglCreateImage().

2.8.1.22 vglDistTransform5()

```
void vglDistTransform5 (
             VglImage * src,
             VglImage * dst,
             VglImage * buf,
             VglImage * 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().

2.8.1.23 vglDistTransformCross3()

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

Distance transform given by elementary cross.

Perform successive erosions 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().

2.8.1.24 vglDistTransformSq3()

```
void vglDistTransformSq3 (
            VglImage * src,
            VglImage * dst,
            VglImage * buf,
            VglImage * 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().

2.8.1.25 vglDownload()

```
void vglDownload (
            VglImage * 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().

2.8.1.26 vglDownloadFaster()

```
void vglDownloadFaster (
            VglImage * 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().

2.8.1.27 vglDownloadFBO()

```
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().

2.8.1.28 vglDownloadPGM()

```
void vglDownloadPGM (
            VglImage * image )
```

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

Use it imediately 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

2.8.1.29 vglDownloadPPM()

```
void vglDownloadPPM (
            VglImage * image )
```

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

Use it imediately 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().

2.8.1.30 vglErodeSq3Sep()

```
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().

2.8.1.31 vglErodeSq5Sep()

```
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().

2.8.1.32 vglGetLevelDistTransform5()

```
void vglGetLevelDistTransform5 (
            VglImage * src,
            VglImage * dst,
            VglImage * buf,
            VglImage * 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().

2.8.1.33 vglHasDisplay()

```
int vglHasDisplay ( )
```

Test if there is display available.

Test if there is display available by checking existence of environment variable DISPLAY.

Referenced by vglInit().

2.8.1.34 vglHorizontalFlip2()

```
void vglHorizontalFlip2 (
            VglImage * src,
            VglImage * dst )
```

Flip image horizontally, that is, left becomes right.

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

2.8.1.35 vglImage3To4Channels()

```
void vglImage3To4Channels (
            VglImage * img )
```

Convert VglImage from 3 to 4 channels

References vglIpl3To4Channels(), and vglNdarray3To4Channels().

2.8.1.36 vglImage4To3Channels()

```
void vglImage4To3Channels (
            VglImage * img )
```

Convert VglImage from 4 to 3 channels

References vglIpl4To3Channels(), and vglNdarray4To3Channels().

2.8.1.37 vglInit() [1/2]

```
int vglInit ( )
```

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

2.8.1.38 vglInit() [2/2]

```
int vglInit (
            int w,
            int h )
```

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

References vglHasDisplay().

2.8.1.39 vglInOut_model()

```
void vglInOut_model (
            VglImage * dst,
            VglImage * dst1 )
```

Test and model for IN_OUT semantics.

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

2.8.1.40 vglIpl3To4Channels()

```
void vglIpl3To4Channels (
            VglImage * img )
```

Convert ipl field of VglImage from 3 to 4 channels

Referenced by vglImage3To4Channels().

2.8.1.41 vglIpl4To3Channels()

```
void vglIpl4To3Channels (
            VglImage * img )
```

Convert ipl field of VglImage from 4 to 3 channels

Referenced by vglImage4To3Channels().

2.8.1.42 vglLoad3dImage()

```
VglImage* vglLoad3dImage (
             char * filename,
             int lStart,
             int lEnd,
             bool has_mipmap )
```

/brief Load sequence of images as 3d image.

Filename must have a printf compatible integer format specifier, like d or %03d.

References vglCreateImage(), and vglReleaseImage().

2.8.1.43 vglLoadImage()

```
VglImage* vglLoadImage (
             char * filename,
             int iscolor,
             int has_mipmap )
```

Load image from file to new VglImage.

This function uses cvLoadImage to read image file.

References vglCreateImage().

2.8.1.44 vglLoadNdImage()

```
VglImage* vglLoadNdImage (
             char * filename,
             int lStart,
             int lEnd,
             int * shape,
             int ndim,
             bool has_mipmap )
```

/brief Load sequence of images as n-dimensional image.

Filename must have a printf compatible integer format specifier, like d or %03d.

References vglCreateImage(), and vglReleaseImage().

2.8.1.45 vglLoadPgm()

```
VglImage* vglLoadPgm (
             char * filename )
```

Load image data from PGM/PPM file.

Load image data from PGM/PPM file, 1 or 3 channels, unsigned byte or short.

References vglCreateImage().

2.8.1.46 vglMultiInput_model()

```
void vglMultiInput_model (
             VglImage * src0,
             VglImage * src1,
             VglImage * dst )
```

Test and model for functions with multiple input images.

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

2.8.1.47 vglMultiOutput_model()

```
void vglMultiOutput_model (
             VglImage * src,
             VglImage * dst,
             VglImage * dst1 )
```

Test and model for functions with multiple output images.

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

2.8.1.48 vglNdarray3To4Channels()

```
void vglNdarray3To4Channels (
             VglImage * img )
```

Converts ndarray from 3 channels to 4 channels

Referenced by vglImage3To4Channels(), and vglUpload().

2.8.1.49 vglNdarray4To3Channels()

```
void vglNdarray4To3Channels (
             VglImage * img )
```

Converts ndarray from 4 channels to 3 channels

Referenced by vglImage4To3Channels().

2.8.1.50 vglOpenSq3()

```
void vglOpenSq3 (
            VglImage * src,
            VglImage * dst,
            VglImage * 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().

2.8.1.51 vglPrintImageData()

```
void vglPrintImageData (
            VglImage * image,
            char * msg,
            char * format )
```

Print image pixels in text format to stdout

2.8.1.52 vglPrintImageInfo()

```
void vglPrintImageInfo (
            VglImage * image,
            char * msg )
```

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 vglClear(), vglDownload(), vglDownloadFaster(), vglDownloadFBO(), and vglUpload().

2.8.1.53 vglReleaseImage()

```
void vglReleaseImage (
            VglImage ** p_image )
```

Release memory occupied by image in RAM and GPU

Referenced by vglLoad3dImage(), and vglLoadNdImage().

2.8.1.54 vglReplaceIpl()

```
void vglReplaceIpl (
            VglImage * image,
            IplImage * new_ipl )
```

Replace IplImage, stored inside a VglImage, 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.

2.8.1.55 vglReshape()

```
int vglReshape (
            VglImage * img,
            VglShape * newShape )
```

/brief Reshape image to given shape.

Reshape image to given shape. Size of original and new shape must be the same.

2.8.1.56 vglSave3dImage()

```
void vglSave3dImage (
            char * filename,
            VglImage * image,
            int lStart,
            int lEnd )
```

Save PGM 3d images on the disk

References vglSaveNdImage().

Referenced by vglSaveImage().

2.8.1.57 vglSaveImage()

```
void vglSaveImage (
            char * filename,
            VglImage * image )
```

Save images with any dimension to disk

TODO: fix 2d.

References vglSave3dImage().

2.8.1.58 vglSaveIplImage()

```
void vglSaveIplImage (
            char * filename,
            IplImage * ipl,
            int * params )
```

Referenced by vglSaveNdImage().

2.8.1.59 vglSaveNdImage()

```
void vglSaveNdImage (
            char * filename,
            VglImage * image,
            int lStart,
            int lEndParam )
```

References vglSaveIplImage().

Referenced by vglSave3dImage().

2.8.1.60 vglSavePgm()

```
int vglSavePgm (
            char * filename,
            VglImage * img )
```

Save image to PGM/PPM file, 1 or 3 channels, unsigned byte

2.8.1.61 vglSavePPM()

```
int vglSavePPM (
            char * filename,
            VglImage * img )
```

Save image to PPM file, 3 channels, unsigned byte

References SavePPM(), and vglDownloadPPM().

2.8.1.62 vglThinBernard()

```
void vglThinBernard (
            VglImage * src,
            VglImage * dst,
            VglImage * 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().

2.8.1.63 vglThinChin()

```
void vglThinChin (
            VglImage * src,
            VglImage * dst,
            VglImage * 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:
```

$$\begin{array}{ccc} P3 & P2 & P1 \\ P4 & P8 & P0 \\ P5 & P6 & P7 \end{array}$$

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().

2.8.1.64 vglUpload()

```
void vglUpload (
            VglImage * 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 vglNdarray3To4Channels(), and vglPrintImageInfo().

2.8.1.65 vglVerticalFlip2()

```
void vglVerticalFlip2 (
            VglImage * src,
            VglImage * dst )
```

Flip image vertically, that is, top becomes bottom.

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

Index