SSESobel

1.0.0

Generated by Doxygen 1.8.7

Mon Jul 25 2016 20:43:19

Contents

Chapter 1

Readme

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

Chapter 5

Class Documentation

5.1 SobelDortmund Class Reference

`#include <SobelDortmund.h>`

Public Types

- enum Direction { Uni, Horizontal, Vertical }

Static Public Member Functions

- static const stdVector2D
  < unsigned char > sobelSSEAnyYUVImageFull (const unsigned char ∗YUVImage, int startX, int startY, int endX, int endY, int width, int height, Direction dir=Uni, bool returnFullArray=true)

  Returns the sobel image for a YUV422 image using every Y value. Corner coordinates are interpreted as image coordinates, which means that if you have a full size image of 1280 by 960, the full size rectangle is defined by (0,0) to (1279, 959) ! Start and end corners can be either top left and bottom right or top right and bottom left in any order.
- static const stdVector2D
  < unsigned char > sobelSSEImageUpperFull (const unsigned char ∗imageUpper, int startX, int startY, int endX, int endY, Direction dir=Uni, bool returnFullArray=true)

  Overloaded function taking the robots upper image instead of any image. A rectangle may be defined.
- static const stdVector2D
  < unsigned char > sobelSSEImageLowerFull (const unsigned char ∗imageLower, int startX, int startY, int endX, int endY, Direction dir=Uni, bool returnFullArray=true)

  Overloaded function taking the robots lower image instead of any image. A rectangle may be defined.
- static const stdVector2D
  < unsigned char > sobelSSEImageUpperFull (const unsigned char ∗imageUpper, Direction dir=Uni)

  Overloaded function taking the robots upper image instead of any image and calculating the sobel operator on the whole image (not a rectangle).
- static const stdVector2D
  < unsigned char > sobelSSEImageLowerFull (const unsigned char ∗imageLower, Direction dir=Uni)

  Overloaded function taking the robots lower image instead of any image and calculating the sobel operator on the whole image (not a rectangle).
- static const stdVector2D
  < unsigned char > sobelSSEAnyYUVImageQuarter (const unsigned char ∗YUVImage, int startX, int startY, int endX, int endY, int width, int height, Direction dir=Uni, bool returnFullArray=true)

  Returns the sobel image for a YUV422 image using every second Y value and every second row. Corner coordinates are interpreted as image coordinates, which means that if you have a quarter size image of 640 by 480, the full size rectangle is defined by (0,0) to (639, 479) ! Start and end corners can be either top left and bottom right or top right and bottom left in any order.

- static const stdVector2D
  < unsigned char > sobelSSEImageUpperQuarter (const unsigned char ∗imageUpper, int startX, int startY, int endX, int endY, Direction dir=Uni, bool returnFullArray=true)

  Overloaded function taking the robots upper image instead of any image. A rectangle and a direction may be defined.
- static const stdVector2D
  < unsigned char > sobelSSEImageUpperQuarter (const unsigned char ∗imageUpper, Direction dir=Uni)

  Overloaded function taking the robots upper image instead of any image and calculating the sobel operator on the whole image (not a rectangle).
- static const stdVector2D
  < unsigned char > sobelSSEImageLowerQuarter (const unsigned char ∗imageLower, int startX, int startY, int endX, int endY, Direction dir=Uni, bool returnFullArray=true)

  Overloaded function taking the robots lower image instead of any image. A rectangle and a direction may be defined.
- static stdVector2D< unsigned char > sobelSSEImageLowerQuarter (const unsigned char ∗imageLower, Direction dir=Uni)

  Overloaded function taking the robots lower image instead of any image and calculating the sobel operator on the whole image (not a rectangle).

5.1.1 Member Enumeration Documentation

5.1.1.1 enum SobelDortmund::Direction

Enumerator

Uni

Horizontal

Vertical

5.1.2 Member Function Documentation

5.1.2.1 const stdVector2D< unsigned char > SobelDortmund::sobelSSEAnyYUVImageFull (const unsigned char ∗ *YUVImage*, int *startX*, int *startY*, int *endX*, int *endY*, int *width*, int *height*, Direction *dir* = Uni, bool *returnFullArray* = `true`) `[static]`

Returns the sobel image for a YUV422 image using every Y value. Corner coordinates are interpreted as image coordinates, which means that if you have a full size image of 1280 by 960, the full size rectangle is defined by (0,0) to (1279, 959) ! Start and end corners can be either top left and bottom right or top right and bottom left in any order.

Parameters

`in`	*YUVImage*	The YUV422 image on which the sobel is calculated.
`in`	*startX*	Start corner in image coordinates, i.e. starting at 0 and ending at width - 1.
`in`	*startY*	Start corner in image coordinates, i.e. starting at 0 and ending at width - 1.
`in`	*endX*	End corner in image coordinates, i.e. starting at 0 and ending at width - 1.
`in`	*endY*	End corner in image coordinates, i.e. starting at 0 and ending at width - 1.
`in`	*width*	Width of the image.
`in`	*height*	Height of the image.
`in`	*dir*	If you want the normal sobel in both horizontal and vertical directions or only one of them. Both (=Uni) is the standard value.
`in`	*returnFullArray*	If you want a full size result even if the defined rectangle is smaller than the full image. Everything outside the rectangle is filled black. Otherwise the result is the size of the rectangle.

Returns

The sobel result.

5.1.2.2 const stdVector2D< unsigned char > SobelDortmund::sobelSSEAnyYUVImageQuarter (const unsigned char ∗ *YUVImage*, int *startX*, int *startY*, int *endX*, int *endY*, int *width*, int *height*, Direction *dir* = Uni, bool *returnFullArray* = `true`) `[static]`

Returns the sobel image for a YUV422 image using every second Y value and every second row. Corner coordinates are interpreted as image coordinates, which means that if you have a quarter size image of 640 by 480, the full size rectangle is defined by (0,0) to (639, 479) ! Start and end corners can be either top left and bottom right or top right and bottom left in any order.

Parameters

`in`	*YUVImage*	The YUV422 image on which the sobel is calculated.
`in`	*startX*	Start corner in image coordinates, i.e. starting at 0 and ending at width - 1.
`in`	*startY*	Start corner in image coordinates, i.e. starting at 0 and ending at width - 1.
`in`	*endX*	End corner in image coordinates, i.e. starting at 0 and ending at width - 1.
`in`	*endY*	End corner in image coordinates, i.e. starting at 0 and ending at width - 1.
`in`	*width*	Width of the quarter image.
`in`	*height*	Height of the quarter image.
`in`	*dir*	If you want the normal sobel in both horizontal and vertical directions or only one of them. Both is standard value.
`in`	*returnFullArray*	If you want a full size result even if the defined rectangle is smaller than the full image. Everything outside the rectangle is filled black. Otherwise the result is the size of the rectangle.

Returns

The sobel result.

5.1.2.3 static const stdVector2D<unsigned char> SobelDortmund::sobelSSEImageLowerFull (const unsigned char ∗ *imageLower*, int *startX*, int *startY*, int *endX*, int *endY*, Direction *dir* = Uni, bool *returnFullArray* = `true`) `[inline]`, `[static]`

Overloaded function taking the robots lower image instead of any image. A rectangle may be defined.

See also

sobelSSEAnyYUVImageFull

5.1.2.4 static const stdVector2D<unsigned char> SobelDortmund::sobelSSEImageLowerFull (const unsigned char ∗ *imageLower*, Direction *dir* = Uni) `[inline]`, `[static]`

Overloaded function taking the robots lower image instead of any image and calculating the sobel operator on the whole image (not a rectangle).

See also

sobelSSEAnyYUVImageFull

5.1.2.5 static const stdVector2D<unsigned char> SobelDortmund::sobelSSEImageLowerQuarter (const unsigned char ∗ *imageLower*, int *startX*, int *startY*, int *endX*, int *endY*, Direction *dir* = Uni, bool *returnFullArray* = `true`) `[inline]`, `[static]`

Overloaded function taking the robots lower image instead of any image. A rectangle and a direction may be defined.

See also

sobelSSEImageUpperQuarter

5.1.2.6 static stdVector2D<unsigned char> SobelDortmund::sobelSSEImageLowerQuarter (const unsigned char ∗ *imageLower*, Direction *dir* = Uni) [inline], [static]

Overloaded function taking the robots lower image instead of any image and calculating the sobel operator on the whole image (not a rectangle).

See also

sobelSSEImageUpperQuarter

5.1.2.7 static const stdVector2D<unsigned char> SobelDortmund::sobelSSEImageUpperFull (const unsigned char ∗ *imageUpper*, int *startX*, int *startY*, int *endX*, int *endY*, Direction *dir* = Uni, bool *returnFullArray* = `true`) [inline], [static]

Overloaded function taking the robots upper image instead of any image. A rectangle may be defined.

See also

sobelSSEAnyYUVImageFull

5.1.2.8 static const stdVector2D<unsigned char> SobelDortmund::sobelSSEImageUpperFull (const unsigned char ∗ *imageUpper*, Direction *dir* = Uni) [inline], [static]

Overloaded function taking the robots upper image instead of any image and calculating the sobel operator on the whole image (not a rectangle).

See also

sobelSSEAnyYUVImageFull

5.1.2.9 static const stdVector2D<unsigned char> SobelDortmund::sobelSSEImageUpperQuarter (const unsigned char ∗ *imageUpper*, int *startX*, int *startY*, int *endX*, int *endY*, Direction *dir* = Uni, bool *returnFullArray* = `true`) [inline], [static]

Overloaded function taking the robots upper image instead of any image. A rectangle and a direction may be defined.

See also

sobelSSEImageUpperQuarter

5.1.2.10 static const stdVector2D<unsigned char> SobelDortmund::sobelSSEImageUpperQuarter (const unsigned char ∗ *imageUpper*, Direction *dir* = Uni) [inline], [static]

Overloaded function taking the robots upper image instead of any image and calculating the sobel operator on the whole image (not a rectangle).

See also

sobelSSEImageUpperQuarter

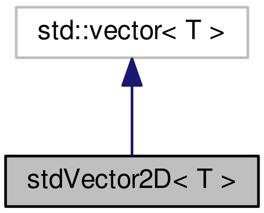
The documentation for this class was generated from the following files:

- include/SobelDortmund.h
- src/SobelDortmund.cpp

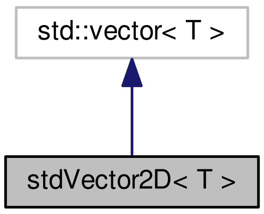
5.2 stdVector2D< T > Class Template Reference

```
#include <Vector2D.h>
```

Inheritance diagram for stdVector2D< T >:



Collaboration diagram for stdVector2D< T >:



Public Member Functions

- stdVector2D (int width, int height)

  Constructor taking width and height of the 2D vector.
- const T & operator() (int x, int y) const

  Can be used to address the data as two dimensional.
- T & operator() (int x, int y)

  Can be used to address the data as two dimensional.

- int getWidth () const

    Returns the width of the 2D vector object.
- int getHeight () const

    Returns the height of the 2D vector object.
- void setHeight (int height)
- void setWidth (int width)

5.2.1 Constructor & Destructor Documentation

5.2.1.1 template<typename T> stdVector2D< T >::stdVector2D (int *width*, int *height*) `[inline]`

Constructor taking width and height of the 2D vector.

Parameters

width	The width.
height	The height.

5.2.2 Member Function Documentation

5.2.2.1 template<typename T> int stdVector2D< T >::getHeight ( ) const `[inline]`

Returns the height of the 2D vector object.

Returns

A copy of the height.

5.2.2.2 template<typename T> int stdVector2D< T >::getWidth ( ) const `[inline]`

Returns the width of the 2D vector object.

Returns

A copy of the width.

5.2.2.3 template<typename T> const T& stdVector2D< T >::operator() (int *x*, int *y*) const `[inline]`

Can be used to address the data as two dimensional.

Parameters

x	X coordinate
y	Y coordinate

Returns

Value at (x,y)

5.2.2.4 template<typename T> T& stdVector2D< T >::operator() (int *x*, int *y*) `[inline]`

Can be used to address the data as two dimensional.

Parameters

x	X coordinate
y	Y coordinate

Returns

Value at (x,y)

5.2.2.5 template<typename T> void **stdVector2D**< T >::setHeight (int *height*) `[inline]`

5.2.2.6 template<typename T> void **stdVector2D**< T >::setWidth (int *width*) `[inline]`

The documentation for this class was generated from the following file:

- include/Vector2D.h

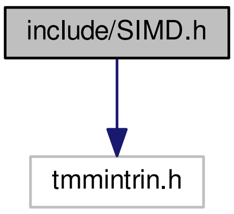
Chapter 6

File Documentation

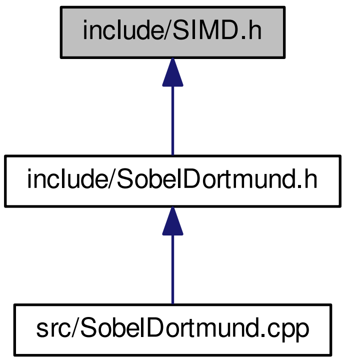
6.1 include/SIMD.h File Reference

```
#include <tmmintrin.h>
```

Include dependency graph for SIMD.h:



This graph shows which files directly or indirectly include this file:



Functions

- __m128i _mm_srli_epi8 (__m128i a, int bits)

  Shifts each of the 16 8-bit integers in a bits right, shifting in zeroes. This function is not available in SSE3, so it emulates the function by copying the register content to two 16-Bit interpreted registers. _mm_srli_epi16 is then called on those two registers and the contents are wrote back to one register.
- __m128i _mm_slli_epi8 (__m128i a, int bits)

  Shifts each of the 16 8-bit integers in a bits left, shifting in zeroes. This function is not available in SSE3, so it emulates the function by copying the register content to two 16-Bit interpreted registers. _mm_srli_epi16 is then called on those two registers and the contents are wrote back to one register.

6.1.1 Detailed Description

Declares some helper functions for SIMD intrinsics

Author

Fabian Rensen

6.1.2 Function Documentation

6.1.2.1 __m128i _mm_slli_epi8 (__m128i *a*, int *bits*) [inline]

Shifts each of the 16 8-bit integers in a bits left, shifting in zeroes. This function is not available in SSE3, so it emulates the function by copying the register content to two 16-Bit interpreted registers. _mm_srli_epi16 is then called on those two registers and the contents are wrote back to one register.

Parameters

in	*a*	SSE Register containing 16 8-bit integers.
in	*bits*	Number of bits to shift the Register a.

Returns

The shifted register

6.1.2.2 __m128i _mm_srli_epi8 (__m128i *a*, int *bits*) [inline]

Shifts each of the 16 8-bit integers in a bits right, shifting in zeroes. This function is not available in SSE3, so it emulates the function by copying the register content to two 16-Bit interpreted registers. _mm_srli_epi16 is then called on those two registers and the contents are wrote back to one register.

Parameters

in	*a*	SSE Register containing 16 8-bit integers.
in	*bits*	Number of bits to shift the Register a.

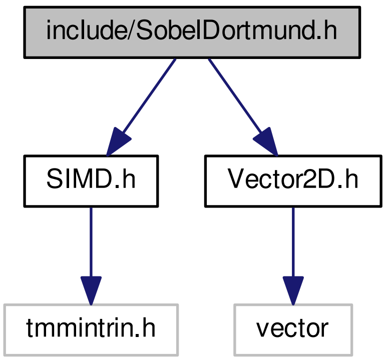
Returns

The shifted register

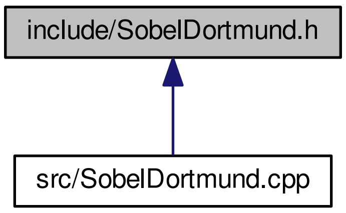
6.2 include/SobelDortmund.h File Reference

```
#include "SIMD.h"
#include "Vector2D.h"
```

Include dependency graph for SobelDortmund.h:



This graph shows which files directly or indirectly include this file:



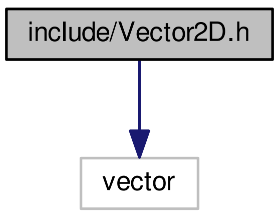
Classes

- class SobelDortmund

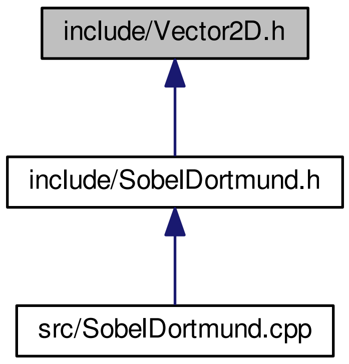
6.3 include/Vector2D.h File Reference

```
#include <vector>
```

Include dependency graph for Vector2D.h:



This graph shows which files directly or indirectly include this file:



Classes

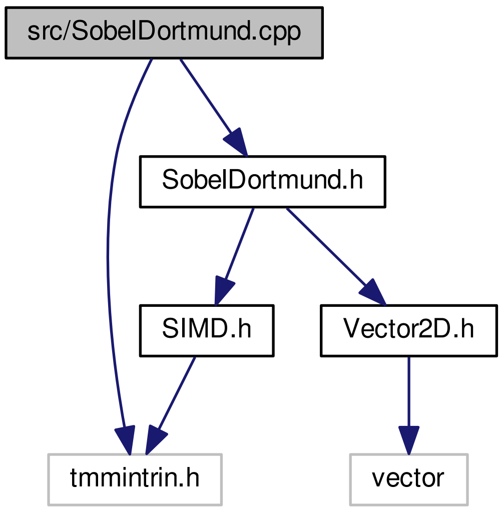
- class stdVector2D< T >

6.4 Readme.md File Reference

6.5 src/SobelDortmund.cpp File Reference

```
#include <tmmintrin.h>
#include "SobelDortmund.h"
```

Include dependency graph for SobelDortmund.cpp:



Index