Visualization

Generated by Doxygen 1.7.3

Mon Nov 21 2011 15:21:37

Contents

1 Namespace Index 1
1.1 Namespace List 1

2 Class Index 3
2.1 Class List 3

3 File Index 5
3.1 File List 5

4 Namespace Documentation 7
4.1 ofv Namespace Reference 7
4.1.1 Detailed Description 8
4.1.2 Typedef Documentation 8
4.1.2.1 data_t 8
4.2 stv Namespace Reference 8
4.2.1 Detailed Description 9
4.2.2 Typedef Documentation 9
4.2.2.1 data_t 9

5 Class Documentation 11
5.1 ofv::CFlowVisualization Class Reference 11
5.1.1 Detailed Description 12
5.1.2 Member Function Documentation 12
5.1.2.1 calcCyclicEncoding 12
5.1.2.2 calcDiffEncoding 13
5.1.2.3 calcDirectionEncoding 13
5.1.2.4 getCycleLength 14
5.1.2.5 getFullSaturationLength 14
5.1.2.6 getSaturationThreshold 14
5.2 stv::CStereoVisualization Class Reference 14
5.2.1 Detailed Description 15
5.2.2 Member Function Documentation 15
5.2.2.1 calcCyclicEncoding 15
5.2.2.2 calcDiffDisparityEncoding 16
5.2.2.3 calcDisparityEncoding 16
5.2.2.4 getCycleLength 17
5.2.2.5 getFullSaturationLength 17
5.2.2.6 getGamma 17

6 File Documentation 19

6.1 flow_visualization.hpp File Reference 19

6.1.1 Detailed Description 20

6.2 stereo_visualization.hpp File Reference 20

6.2.1 Detailed Description 22

Chapter 1

Namespace Index

1.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

ofv (All classes and functions to visualize optical flow fields) 7
stv (All classes and functions to visualize stereo disparities) 8

Chapter 2

Class Index

2.1 Class List

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

ofv::CFlowVisualization (Optical Flow Visualization) 11
stv::CStereoVisualization (Stereo Visualization) 14

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

flow_visualization.hpp (Header for visualization of optical flow) 19
stereo_visualization.hpp (Header for visualization of stereo disparities) . . . 20

Chapter 4

Namespace Documentation

4.1 ofv Namespace Reference

All classes and functions to visualize optical flow fields.

Classes

- class CFlowVisualization

  Optical Flow Visualization.

Typedefs

- typedef float data_t

  Throughout the visualizations, the given flow field is assumed to be of datatype data_t.

Functions

- data_t getValidMean (const ::cimg_library::CImg< data_t > &f_I_r, const ::cimg_library::CImg< bool > &f_valid_r)

  From a given image and validity-matrix determine the mean of all valid entries.

Variables

- const data_t COLOR_OFFSET = static_cast<data_t> (7.0 / 16.0)

  Shift colour map so that downward motion is yellow.

- const int N_COLORS = 128

Number of colours in the map.

4.1.1 Detailed Description

All classes and functions to visualize optical flow fields.

> This namespace contains - apart from the CFlowVisualization class - some useful helper functions and constants.

4.1.2 Typedef Documentation

4.1.2.1 typedef float ofv::data_t

Throughout the visualizations, the given flow field is assumed to be of datatype data_t.

Attention: data_t has to be a floating type!

4.2 stv Namespace Reference

All classes and functions to visualize stereo disparities.

Classes

- class CStereoVisualization

  Stereo Visualization.

Typedefs

- typedef float data_t

  Throughout the visualizations, the given disparity is assumed to be of datatype data_t.

Functions

- data_t getValidMin (const ::cimg_library::CImg< data_t > &f_I_r, const ::cimg_library::CImg< bool > &f_valid_r)

  From a given image and validity-matrix determine the min of all valid entries.

- data_t getValidMax (const ::cimg_library::CImg< data_t > &f_I_r, const ::cimg_library::CImg< bool > &f_valid_r)

  From a given image and validity-matrix determine the max of all valid entries.

Variables

- const data_t COLOR_OFFSET = static_cast<data_t>(2.0 / 6.0)

  <shift so that zero disparity is blue

- const int N_COLORS = 128

  Number of colours in the map.

4.2.1 Detailed Description

All classes and functions to visualize stereo disparities.

> This namespace contains - apart from the CStereoVisualization class - some useful helper functions and constants.

4.2.2 Typedef Documentation

4.2.2.1 typedef float stv::data_t

Throughout the visualizations, the given disparity is assumed to be of datatype data_t.

Attention: data_t has to be a floating type!

Chapter 5

Class Documentation

5.1 ofv::CFlowVisualization Class Reference

Optical Flow Visualization.

`#include <flow_visualization.hpp>`

Public Member Functions

- CFlowVisualization ()

  Default Constructor. Sets the standard parameter and initializes the look-up table.

- ~CFlowVisualization ()

  Default Destructor.

- void setSaturationThreshold (const double f_saturationThreshold)

  Components longer than this threshold are compressed to avoid oversaturation in the direction encoding.

- double getSaturationThreshold ()

  Returns the threshold for compression in the direction encoding.

- void setFullSaturationLength (const int f_fullSaturationLength)

  At FullSaturationLength the motion is fully saturated in the direction encoding.

- int getFullSaturationLength ()

  Returns the length at which the colour of the direction encoding is fully saturated.

- void setCycleLength (const int f_cycleLength)

  CycleLength is the pixel-displacement that fits into one cycle of colours in the cyclic representation.

- int getCycleLength ()

  Returns the number of pixels after which the cyclic colour encoding repeats itself.

- void calcDirectionEncoding (::cimg_library::CImg< unsigned char > &f_rgbImage_r, const ::cimg_library::CImg< data_t > &f_flowU_r, const ::cimg_library::CImg< data_t > &f_flowV_r, const ::cimg_library::CImg< bool > &f_valid_r) const

  Determine colour and saturation based on motion direction and length.

- void calcCyclicEncoding (::cimg_library::CImg< unsigned char > &f_rgbImage_r, const ::cimg_library::CImg< data_t > &f_flowU_r, const ::cimg_library::CImg< data_t > &f_flowV_r, const ::cimg_library::CImg< bool > &f_valid_r) const

  Determine cyclic colour values based on the length of the motion.

- void calcDiffEncoding (::cimg_library::CImg< unsigned char > &f_rgbImage_r, const ::cimg_library::CImg< data_t > &f_flowU_r, const ::cimg_library::CImg< data_t > &f_flowV_r, const ::cimg_library::CImg< bool > &f_valid_r, const data_t diffU, const data_t diffV) const

  Determine colour and saturation based on a (mean-)adjusted motion field.

5.1.1 Detailed Description

Optical Flow Visualization. Getter and Setter functions for the visualization parameters are provided.

This class provides the colour map and the functions for the visualization of optical flow as they are used on the webpage

`http://hci.iwr.uni-heidelberg.de/Benchmarks/`

However, please indicate clearly whenever parameters for the visualizations are changed!

5.1.2 Member Function Documentation

5.1.2.1 void ofv::CFlowVisualization::calcCyclicEncoding (::cimg_library::CImg< unsigned char > & *f_rgbImage_r*, const ::cimg_library::CImg< data_t > & *f_flowU_r*, const ::cimg_library::CImg< data_t > & *f_flowV_r*, const ::cimg_library::CImg< bool > & *f_valid_r*) const

Determine cyclic colour values based on the length of the motion.

Fill an RGB colour image that visualizes the input flow and its validity. This visualization is directionally insensitive but highly sensitive to variations in the length of motion vectors. It allows to see small length variations within presumely constant regions clearly.

Parameters

`in,out`	*f_rgbImage_r*	reference to a preallocated RGB-image that receives the determined color values

in	*f_flowU_r*	reference to the horizontal flow component
in	*f_flowV_r*	reference to the vertical flow component
in	*f_valid_r*	reference to the image indicating the validity of the determined flow values.

5.1.2.2 void ofv::CFlowVisualization::calcDiffEncoding (::cimg_library::CImg< unsigned char > & *f_rgbImage_r*, const ::cimg_library::CImg< data_t > & *f_flowU_r*, const ::cimg_library::CImg< data_t > & *f_flowV_r*, const ::cimg_library::CImg< bool > & *f_valid_r*, const data_t *diffU*, const data_t *diffV*) const

Determine colour and saturation based on a (mean-)adjusted motion field.

The content of motion field is adjusted horizontally and vertically by the provided amount and than a direction and length dependend colour encoding that is filled into an RGB image. This visualization allows to shift the clearly distinguishable, non-saturated range from (0,0) to any desired range. A common variant is to adjust the motion field by the mean of each component. A function to determine the mean value of all valid flow components is provided with ofv::getValidMean

Parameters

in,out	*f_rgbImage_r*	reference to a preallocated RGB-image that receives the determined color values
in	*f_flowU_r*	reference to the horizontal flow component
in	*f_flowV_r*	reference to the vertical flow component
in	*f_valid_r*	reference to the image indicating the validity of the determined flow values.
in	*diffU*	the amount by which the horizontal component is adjusted.
in	*diffV*	the amount by which the vertical component is adjusted.

5.1.2.3 void ofv::CFlowVisualization::calcDirectionEncoding (::cimg_library::CImg< unsigned char > & *f_rgbImage_r*, const ::cimg_library::CImg< data_t > & *f_flowU_r*, const ::cimg_library::CImg< data_t > & *f_flowV_r*, const ::cimg_library::CImg< bool > & *f_valid_r*) const

Determine colour and saturation based on motion direction and length.

Fill an RGB colour image that visualizes the input flow and its validity. This customary visualization where direction is expressed by the colour and length by the saturation allows to see directional outliers quickly

Parameters

in,out	*f_rgbImage_r*	reference to a preallocated RGB-image that receives the determined color values
in	*f_flowU_r*	reference to the horizontal flow component
in	*f_flowV_r*	reference to the vertical flow component
in	*f_valid_r*	reference to the image indicating the validity of the determined flow values

5.1.2.4 int ofv::CFlowVisualization::getCycleLength ( ) `[inline]`

Returns the number of pixels after which the cyclic colour encoding repeats itself.

Whenever this threshold is modified, the value of the threshold should be provided.

5.1.2.5 int ofv::CFlowVisualization::getFullSaturationLength ( ) `[inline]`

Returns the length at which the colour of the direction encoding is fully saturated.

Whenever this threshold is modified, the value of the threshold should be provided.

5.1.2.6 double ofv::CFlowVisualization::getSaturationThreshold ( ) `[inline]`

Returns the threshold for compression in the direction encoding.

Whenever this threshold is modified, the value of the threshold should be provided.

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

- flow_visualization.hpp

5.2 stv::CStereoVisualization Class Reference

Stereo Visualization.

```
#include <stereo_visualization.hpp>
```

Public Member Functions

- CStereoVisualization ()

  Default Constructor. Sets the standard parameter and initializes the look-up table.

- ~CStereoVisualization ()

  Default Destructor.

- void setGamma (const double f_gamma)

  Input disparities are compressed by exponentiation with gamma<=1.

- double getGamma ()

  Returns the compression exponent gamma.

- void setFullSaturationLength (const int f_fullSaturationLength)

  For the basic non-scaled colour encoding this value is the maximal, uniquely representable disparity.

- int getFullSaturationLength ()

Returns the maximal, uniquely representable disparity.

- void setCycleLength (const int f_cycleLength)

  CycleLength is the pixel-displacement that fits into one cycle of colours in the cyclic representation.

- int getCycleLength ()

  Returns the number of pixels after which the cyclic colour encoding repeats itself.

- void calcDisparityEncoding (::cimg_library::CImg< unsigned char > &f_rgbImage_r, const ::cimg_library::CImg< data_t > &f_disparity_r, const ::cimg_library::CImg< bool > &f_valid_r) const

  Determine a unique colour based on the input disparity and the member fullSaturationLength.

- void calcCyclicEncoding (::cimg_library::CImg< unsigned char > &f_rgbImage_r, const ::cimg_library::CImg< data_t > &f_disparity_r, const ::cimg_library::CImg< bool > &f_valid_r) const

  Determine cyclic colour values based on the length of the disparity.

- void calcDiffDisparityEncoding (::cimg_library::CImg< unsigned char > &f_rgbImage_r, const ::cimg_library::CImg< data_t > &f_disparity_r, const ::cimg_library::CImg< bool > &f_valid_r, const data_t f_minDisparity, const data_t f_maxDisparity) const

  For disparity between the lower and upper limit determine a unique colour value.

5.2.1 Detailed Description

Stereo Visualization. Getter and Setter functions for the visualization parameters are provided.

This class provides the colour map and the functions for the visualization of stereo disparity maps as they are used on the webpage

`http://hci.iwr.uni-heidelberg.de/Benchmarks/`

However, please indicate clearly whenever parameters for the visualizations are changed!

5.2.2 Member Function Documentation

5.2.2.1 void stv::CStereoVisualization::calcCyclicEncoding (::cimg_library::CImg< unsigned char > & *f_rgbImage_r*, const ::cimg_library::CImg< data_t > & *f_disparity_r*, const ::cimg_library::CImg< bool > & *f_valid_r*) const

Determine cyclic colour values based on the length of the disparity.

Fill an RGB colour image that visualizes the input disparity and its validity. Repeating the available colour cyclically. This visualization allows to see small vairiations within presumely constant regions clearly. This visualization allows to compare different algorithms with different ranges of resulting disparities.

Parameters

in,out	*f_rgbImage_r*	reference to a preallocated RGB-image that receives the determined color values
in	*f_disparity_r*	reference to the stereo disparity
in	*f_valid_r*	reference to the image indicating the validity of the determined disparity values.

5.2.2.2 void stv::CStereoVisualization::calcDiffDisparityEncoding (::cimg_library::CImg< unsigned char > & *f_rgbImage_r*, const ::cimg_library::CImg< data_t > & *f_disparity_r*, const ::cimg_library::CImg< bool > & *f_valid_r*, const data_t *f_minDisparity*, const data_t *f_maxDisparity*) const

For disparity between the lower and upper limit determine a unique colour value.

Fill an RGB colour image that visualizes the input disparity and its validity. For disparity values between lower and upper limit this colour code is unique. Outside these limits colour repeat themselves. Stretching the disparity between the limits allows to fully exploit the colour spectrum. If the limits are chose to be the min and max of the current disparity field, comparability to other algorithms is not given. For comparable visualizations see stv::CStereoVisualization::calcDisparityEncoding() where standard-limits are provided.

Parameters

in,out	*f_rgbImage_r*	reference to a preallocated RGB-image that receives the determined colour values
in	*f_disparity_r*	reference to the stereo disparity
in	*f_valid_r*	reference to the image indicating the validity of the determined disparity values.
in	*f_minDisparity*	the smallest uniquely displayed disparity.
in	*f_maxDisparity*	the larges uniquely displayed disparity.

5.2.2.3 void stv::CStereoVisualization::calcDisparityEncoding (::cimg_library::CImg< unsigned char > & *f_rgbImage_r*, const ::cimg_library::CImg< data_t > & *f_disparity_r*, const ::cimg_library::CImg< bool > & *f_valid_r*) const

Determine a unique colour based on the input disparity and the member fullSaturationLength.

Fill an RGB colour image that visualizes the input disparity and its validity. For disparity values between 0 and fullSaturationLength this colour code is unique. Outside these limits colour repeat themselves. Using stv::CStereoVisualization::calcDiffDisparityEncoding() upper and lower limits can be set manually. This standardised visualization allows to compare different algorithms with different ranges of resulting disparities.

Parameters

in,out	*f_rgbImage_r*	reference to a preallocated RGB-image that receives the determined colour values
in	*f_disparity_r*	reference to the stereo disparity
in	*f_valid_r*	reference to the image indicating the validity of the determined disparity values.

5.2.2.4 int stv::CStereoVisualization::getCycleLength ( ) [inline]

Returns the number of pixels after which the cyclic colour encoding repeats itself.

Whenever this threshold is modified, the value of the threshold should be provided.

5.2.2.5 int stv::CStereoVisualization::getFullSaturationLength ( ) [inline]

Returns the maximal, uniquely representable disparity.

Whenever this threshold is modified, the value of the threshold should be provided.

5.2.2.6 double stv::CStereoVisualization::getGamma ( ) [inline]

Returns the compression exponent gamma.

Whenever the exponent is modified, the value of gamma should be provided.

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

- stereo_visualization.hpp

Chapter 6

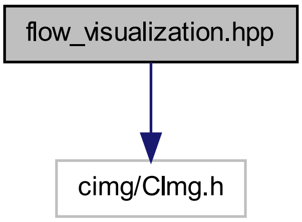
File Documentation

6.1 flow_visualization.hpp File Reference

Header for visualization of optical flow.

```
#include "cimg/CImg.h"
```

Include dependency graph for flow_visualization.hpp:



Classes

- class ofv::CFlowVisualization

  Optical Flow Visualization.

Namespaces

- namespace ofv

All classes and functions to visualize optical flow fields.

Typedefs

- typedef float ofv::data_t

  Throughout the visualizations, the given flow field is assumed to be of datatype data_t.

Functions

- data_t ofv::getValidMean (const ::cimg_library::CImg< data_t > &f_I_r, const ::cimg_library::CImg< bool > &f_valid_r)

  From a given image and validity-matrix determine the mean of all valid entries.

Variables

- const data_t ofv::COLOR_OFFSET = static_cast<data_t> (7.0 / 16.0)

  Shift colour map so that downward motion is yellow.

- const int ofv::N_COLORS = 128

  Number of colours in the map.

6.1.1 Detailed Description

Header for visualization of optical flow.

Synopsis:

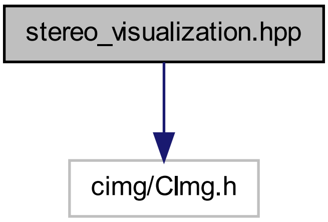
This file contains the functions and classes for the visualization of optical flow.

6.2 stereo_visualization.hpp File Reference

Header for visualization of stereo disparities.

```
#include "cimg/CImg.h"
```

Include dependency graph for stereo_visualization.hpp:



Classes

- class stv::CStereoVisualization

  Stereo Visualization.

Namespaces

- namespace stv

  All classes and functions to visualize stereo disparities.

Typedefs

- typedef float stv::data_t

  Throughout the visualizations, the given disparity is assumed to be of datatype data_t.

Functions

- data_t stv::getValidMin (const ::cimg_library::CImg< data_t > &f_I_r, const ::cimg_library::CImg< bool > &f_valid_r)

  From a given image and validity-matrix determine the min of all valid entries.

- data_t stv::getValidMax (const ::cimg_library::CImg< data_t > &f_I_r, const ::cimg_library::CImg< bool > &f_valid_r)

  From a given image and validity-matrix determine the max of all valid entries.

Variables

- const data_t stv::COLOR_OFFSET = static_cast<data_t>(2.0 / 6.0)

  <shift so that zero disparity is blue

- const int stv::N_COLORS = 128

  Number of colours in the map.

6.2.1 Detailed Description

Header for visualization of stereo disparities.

Synopsis:

> This file contains the functions and classes for the visualization of stereo disparities.