Ambient Light Transfer

Generated by Doxygen 1.7.6.1

Sun Sep 29 2013 22:25:36

Contents

1 Data Structure Index

1.1 Class Hierarchy

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

2 Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

3 File Index

3.1 File List

Here is a list of all files with brief descriptions:

4 Data Structure Documentation

4.1 Calibrate Class Reference

The Calibration Loop.

`#include <calibrate.h>`

Data Structures

- struct params

  Configuration of Calibrate class.

Public Member Functions

- Calibrate (Lightprobe ∗p, Lamps ∗l, params c)
- Calibrate (Lightprobe ∗p, Lamps ∗l, string path, double rate)
- ∼Calibrate ()
- params getConfig ()
- void runCaptureImpacts ()
- void runCalibrateLamps ()

Private Attributes

- params config
- Lightprobe ∗ probe
- Lamps ∗ lamps

4.1.1 Detailed Description

The Calibration Loop.

Author

Manuel Jerger <nom@nomnom.de> The Calibration Loop for use with the webcam probe. Can also measure the response curve of our lighting system.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 Calibrate::Calibrate (Lightprobe ∗ *p*, Lamps ∗ *l*, params *c*)

Author

Manuel Jerger <nom@nomnom.de>

The Calibration Loop for use with the webcam probe. Can also measure the response curve of our lighting system.

4.1.2.2 Calibrate::Calibrate (Lightprobe ∗ *p*, Lamps ∗ *l*, string *path*, double *rate*)

4.1.2.3 Calibrate::∼Calibrate ( )

4.1.3 Member Function Documentation

4.1.3.1 Calibrate::params Calibrate::getConfig ( )

4.1.3.2 void Calibrate::runCalibrateLamps ( )

Run the Lamp-Calibration and calculate LED response using an image source and prerecorded images.

4.1.3.3 void Calibrate::runCaptureImpacts ( )

Run the calibration loop using the webcam.

4.1.4 Field Documentation

4.1.4.1 params Calibrate::config `[private]`

4.1.4.2 Lamps∗ Calibrate::lamps `[private]`

4.1.4.3 Lightprobe∗ Calibrate::probe `[private]`

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

- src/calibrate.h
- src/calibrate.cpp

4.2 Calibrate::params Struct Reference

Configuration of Calibrate class.

`#include <calibrate.h>`

Public Member Functions

- params ()

Data Fields

- string dataDir
- double captureRate

4.2.1 Detailed Description

Configuration of Calibrate class.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 Calibrate::params::params ( ) `[inline]`

4.2.3 Field Documentation

4.2.3.1 double Calibrate::params::captureRate

4.2.3.2 string Calibrate::params::dataDir

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

- src/calibrate.h

4.3 circle Struct Reference

A circle.

`#include <utils.h>`

Public Member Functions

- circle (double x, double y, double r)
- circle ()
- bool isValid ()

Data Fields

- double x
- double y
- double r

4.3.1 Detailed Description

A circle.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 circle::circle (double *x*, double *y*, double *r*) `[inline]`

4.3.2.2 circle::circle ( ) `[inline]`

4.3.3 Member Function Documentation

4.3.3.1 bool circle::isValid ( ) `[inline]`

4.3.4 Field Documentation

4.3.4.1 double circle::r

4.3.4.2 double circle::x

4.3.4.3 double circle::y

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

- src/utils.h

4.4 dirCone Struct Reference

Stores the neighborhood of one sampling direction.

```
#include <utils.h>
```

Public Member Functions

- dirCone ()

- dirCone (Vector3d dir)
- void add (Vector3d dir, Vector2i pixel, double weight)

Data Fields

- Vector3d direction
- directions dirs
- vector< Vector2i > pixels
- vector< double > weights
- double weightSum

4.4.1 Detailed Description

Stores the neighborhood of one sampling direction.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 dirCone::dirCone ( ) `[inline]`

4.4.2.2 dirCone::dirCone (Vector3d *dir*) `[inline]`

4.4.3 Member Function Documentation

4.4.3.1 void dirCone::add (Vector3d *dir*, Vector2i *pixel*, double *weight*) `[inline]`

4.4.4 Field Documentation

4.4.4.1 Vector3d dirCone::direction

4.4.4.2 directions dirCone::dirs

4.4.4.3 vector<Vector2i> dirCone::pixels

4.4.4.4 vector<double> dirCone::weights

4.4.4.5 double dirCone::weightSum

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

- src/utils.h

4.5 Gui Class Reference

The user interface.

```
#include <gui.h>
```

Public Member Functions

- Gui ()
- Gui (int width, int height)
- ∼Gui ()
- void start (int width, int height)
- const void drawLine (point f, point t)
- const void drawLine (int x1, int y1, int x2, int y2)
- const void drawBox (rect box)
- const void drawBox (int x1, int y1, int x2, int y2)
- const void drawCircle (circle c)
- const void drawCircle (circle c, int x, int y)
- const void drawCircleFilled (int xpos, int ypos, int radius)
- const void drawPoints (vector< Vector2i >)
- const void drawPoints (vector< Vector2i >, int x, int y)
- const void drawImage (imglib::Image< float > &, int, int)
- const void drawImage (imglib::Image< float > &img, int xpos, int ypos, bool isLinear)
- const void drawCircularImage (imglib::Image< float > &img, circle, int x, int y)
- const void drawCircularImage (imglib::Image< float > &, circle)
- const void drawStickRGBGrid (Lamps ∗, int x, int y, int w, int h, int space, bool)
- const void drawMonochromeLamps (Lamps ∗lamps, int x, int y, int box_width, int box_height, int space, bool vertical)
- const circle runSphereSelection (Source ∗)
- const point runPixelSelection (Source ∗)

4.5.1 Detailed Description

The user interface.

Author

Manuel Jerger <`nom@nomnom.de`>

The graphic user interface. Uses OpenGL via the glfw abstraction layer.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 Gui::Gui ( ) `[inline]`

4.5.2.2 Gui::Gui (int *width*, int *height*)

4.5.2.3 Gui::∼Gui ( )

4.5.3 Member Function Documentation

4.5.3.1 const void Gui::drawBox (rect *box*)

4.5.3.2 const void **Gui::drawBox** (int *x1,* int *y1,* int *x2,* int *y2*)

4.5.3.3 const void **Gui::drawCircle** (**circle** *c*)

4.5.3.4 const void **Gui::drawCircle** (**circle** *c,* int *x,* int *y*)

4.5.3.5 const void **Gui::drawCircleFilled** (int *xpos,* int *ypos,* int *radius*)

4.5.3.6 const void **Gui::drawCircularImage** (imglib::Image< float > & *img,* **circle** *area,* int *x,* int *y*)

4.5.3.7 const void **Gui::drawCircularImage** (imglib::Image< float > & *img,* **circle** *area*)

4.5.3.8 const void **Gui::drawImage** (imglib::Image< float > & *img,* int *xpos,* int *ypos*)

4.5.3.9 const void **Gui::drawImage** (imglib::Image< float > & *img,* int *xpos,* int *ypos,* bool *isLinear*)

4.5.3.10 const void **Gui::drawLine** (**point** *f,* **point** *t*)

4.5.3.11 const void **Gui::drawLine** (int *x1,* int *y1,* int *x2,* int *y2*)

4.5.3.12 const void **Gui::drawMonochromeLamps** (**Lamps** ∗ *lamps,* int *x,* int *y,* int *box_width,* int *box_height,* int *space,* bool *vertical*)

Draw value of monochrome lamps as white rectangle.

4.5.3.13 const void **Gui::drawPoints** (vector< Vector2i > *points*)

4.5.3.14 const void **Gui::drawPoints** (vector< Vector2i > *points,* int *x,* int *y*)

4.5.3.15 const void **Gui::drawStickRGBGrid** (**Lamps** ∗ *lamps,* int *x,* int *y,* int *box_width,* int *box_height,* int *space,* bool *vertical*)

Draw the lamp's RGB values as a table of rectangles

Parameters

x	Top left position inside window.
y	Top left position inside window.
box_width	Width of rectangle.
box_height	Height of rectangle.
space	Spacing between rectangles.
vertical	Orientation

4.5.3.16 const **point Gui::runPixelSelection** (**Source** ∗ *source*)

Displays images acquired from source and asks user to select a pixel / 2D coordinate.

Parameters

source	Image source.

Returns

The pixel position.

4.5.3.17 const circle Gui::runSphereSelection (Source ∗ *source*)

Displays images acquired from source and asks user to select a sphere by clicking three points on the border of a circle.

Parameters

source	Image source.

Returns

The parameters of the circle.

4.5.3.18 void Gui::start (int *width*, int *height*)

Start the GUI with a specified window size. Sets up OpenGL and displays the window.

Parameters

width	Width of window in pixel.
height	Height of window in pixel.

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

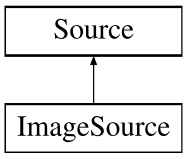
- src/gui.h
- src/gui.cpp

4.6 ImageSource Class Reference

Source that uses image files.

```
#include <imagesource.h>
```

Inheritance diagram for ImageSource:



Data Structures

- struct params

  Configuration of the ImageSource class.

Public Member Functions

- ImageSource (params c)
- ∼ImageSource ()
- params getConfig ()
- bool hasNewData ()
- void acquire ()

Private Attributes

- params config
- int curImageID

4.6.1 Detailed Description

Source that uses image files.

Author

Manuel Jerger <`nom@nomnom.de`>

Specialization of the source class that uses image files. Loads either a single image file or a whole sequence of frames from a directory.

4.6.2 Constructor & Destructor Documentation

4.6.2.1 ImageSource::ImageSource (params *c*)

Author

Manuel Jerger <`nom@nomnom.de`>

Specialization of the source class that uses image files. Loads either a single image file or a whole sequence of frames from a directory. Constructor checks if config specifies a single image (ending in .ppm) or a directory. It then loads the first image to check for the dimensions.

4.6.2.2 ImageSource::∼ImageSource ( ) [inline]

4.6.3 Member Function Documentation

4.6.3.1 void ImageSource::acquire ( ) [virtual]

Loads either a single image or a whole directory of frames. The latter one requires the files to be named img_#.ppm where # is a number w/0 trailing zeroes.

Implements Source.

4.6.3.2 ImageSource::params ImageSource::getConfig ( )

4.6.3.3 bool ImageSource::hasNewData ( ) [virtual]

Implements Source.

4.6.4 Field Documentation

4.6.4.1 params ImageSource::config [private]

4.6.4.2 int ImageSource::curImageID [private]

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

- src/imagesource.h
- src/imagesource.cpp

4.7 ImageSource::params Struct Reference

Configuration of the ImageSource class.

`#include <imagesource.h>`

Public Member Functions

- params ()

Data Fields

- string imagePath

4.7.1 Detailed Description

Configuration of the ImageSource class.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 ImageSource::params::params ( ) `[inline]`

4.7.3 Field Documentation

4.7.3.1 string ImageSource::params::imagePath

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

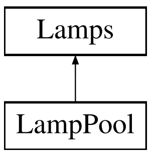
- src/imagesource.h

4.8 LampPool Class Reference

Groups instances of Lamps.

`#include <lamppool.h>`

Inheritance diagram for LampPool:



Public Member Functions

- LampPool ()
- ~LampPool ()
- int getNumLamps ()
- void setValue (int, double)
- double getValue (int)
- void setAll (double)
- bool doStep (double delta_t)
- void setFadeSpeed (double speed)
- void setUpdateRate (double rate)
- void start ()
- void stop ()
- bool isReady ()
- bool send ()
- int getNumMembers ()
- Lamps ∗ getMember (int m)
- void addMember (Lamps ∗lamps)

Private Member Functions

- int getMemberForLampIndex (int)
- int getMappedLampIndex (int)

Private Attributes

- vector< Lamps ∗ > members

4.8.1 Detailed Description

Groups instances of Lamps.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 LampPool::LampPool ( )

Author

Manuel Jerger <`nom@nomnom.de`>

This class represents a pool of lamps. It controls multiple Lamps classes and behaves like a single lamp class.

4.8.2.2 LampPool::∼LampPool ( ) `[inline]`

4.8.3 Member Function Documentation

4.8.3.1 void LampPool::addMember (Lamps ∗ *lamps*)

4.8.3.2 bool LampPool::doStep (double *delta_t*) `[virtual]`

Does one fading step (if fadespeed > 0) and calls send() at the end.

Reimplemented from Lamps.

4.8.3.3 int LampPool::getMappedLampIndex (int *lampID*) `[private]`

4.8.3.4 Lamps ∗ LampPool::getMember (int *m*)

4.8.3.5 int LampPool::getMemberForLampIndex (int *lampID*) `[private]`

4.8.3.6 int LampPool::getNumLamps ( ) `[virtual]`

Reimplemented from Lamps.

4.8.3.7 int LampPool::getNumMembers ( )

4.8.3.8 double LampPool::getValue (int *lampID*) `[virtual]`

Return the current brightness of a single lamp.

Reimplemented from Lamps.

4.8.3.9 bool LampPool::isReady ( ) [virtual]

Implements Lamps.

4.8.3.10 bool LampPool::send ( ) [virtual]

Implements Lamps.

4.8.3.11 void LampPool::setAll (double *brightness*) [virtual]

Set all lamps to the specified brightness.

Reimplemented from Lamps.

4.8.3.12 void LampPool::setFadeSpeed (double *speed*) [virtual]

Reimplemented from Lamps.

4.8.3.13 void LampPool::setUpdateRate (double *rate*) [virtual]

Reimplemented from Lamps.

4.8.3.14 void LampPool::setValue (int *lampID,* double *brightness*) [virtual]

Sets the brightness of a lamp. If fading is enabled, it sets the target fade-to value.

Reimplemented from Lamps.

4.8.3.15 void LampPool::start ( )

Starts fading thread for automatic updating and fading.

Reimplemented from Lamps.

4.8.3.16 void LampPool::stop ( )

Stops automatic updating.

Reimplemented from Lamps.

4.8.4 Field Documentation

4.8.4.1 vector<Lamps*> LampPool::members [private]

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

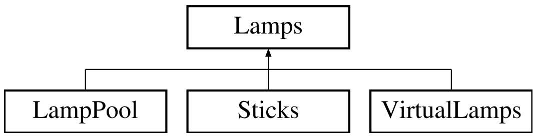
- src/lamppool.h
- src/lamppool.cpp

4.9 Lamps Class Reference

A monochrome lamp.

`#include <lamps.h>`

Inheritance diagram for Lamps:



Public Member Functions

- Lamps ()
- ∼Lamps ()
- virtual int getNumLamps ()
- virtual void setValue (int, double)
- virtual double getValue (int)
- virtual void setAll (double)
- void start ()
- void stop ()
- virtual bool doStep (double delta_t)
- virtual void setFadeSpeed (double speed)
- virtual void setUpdateRate (double rate)
- virtual bool isReady ()=0
- virtual bool send ()=0

Static Protected Member Functions

- static void ∗ start_thread (void ∗ptr)

Protected Attributes

- bool running
- int numLamps
- vector< double > lampValues
- vector< double > previousLampValues
- vector< double > lampTargetValues
- double fadeSpeed
- double updateRate

4.9.1 Detailed Description

A monochrome lamp.

4.9.2 Constructor & Destructor Documentation

4.9.2.1 Lamps::Lamps ( )

Author

Manuel Jerger <`nom@nomnom.de`>

This abstract class represents a group of lamps of identical type (controlled by the same hardware). A lamp is monochrome and takes a rational value from 0..1 , where 1.0 is the maximum brightness and 0 turns the lamp off. It provides virtuals for setting lamp values, driving the hardware, automatic fading. It provides threaded updating.

4.9.2.2 Lamps::∼Lamps ( )

4.9.3 Member Function Documentation

4.9.3.1 bool Lamps::doStep (double *delta_t*) `[virtual]`

Does one fading step (if fadespeed > 0) and calls send() at the end.

Reimplemented in LampPool.

4.9.3.2 int Lamps::getNumLamps ( ) `[virtual]`

Reimplemented in LampPool.

4.9.3.3 double Lamps::getValue (int *lampID*) `[virtual]`

Return the current brightness of a single lamp.

Reimplemented in LampPool.

4.9.3.4 virtual bool Lamps::isReady ( ) `[pure virtual]`

Implemented in Sticks, LampPool, and VirtualLamps.

4.9.3.5 virtual bool Lamps::send ( ) `[pure virtual]`

Implemented in Sticks, LampPool, and VirtualLamps.

4.9.3.6 void Lamps::setAll (double *brightness*) `[virtual]`

Set all lamps to the specified brightness.

Reimplemented in LampPool.

4.9.3.7 void Lamps::setFadeSpeed (double *speed*) `[virtual]`

Reimplemented in LampPool.

4.9.3.8 void Lamps::setUpdateRate (double *rate*) `[virtual]`

Reimplemented in LampPool.

4.9.3.9 void **Lamps::setValue** (int *lampID,* double *brightness*) `[virtual]`

Sets the brightness of a lamp. If fading is enabled, it sets the target fade-to value.

Reimplemented in LampPool.

4.9.3.10 void **Lamps::start** ( )

Starts fading thread for automatic updating and fading.

Reimplemented in LampPool.

4.9.3.11 void ∗ **Lamps::start_thread** (void ∗ *ptr*) `[static, protected]`

The updating thread.

4.9.3.12 void **Lamps::stop** ( )

Stops automatic updating.

Reimplemented in LampPool.

4.9.4 Field Documentation

4.9.4.1 double **Lamps::fadeSpeed** `[protected]`

4.9.4.2 vector<double> **Lamps::lampTargetValues** `[protected]`

4.9.4.3 vector<double> **Lamps::lampValues** `[protected]`

4.9.4.4 int **Lamps::numLamps** `[protected]`

4.9.4.5 vector<double> **Lamps::previousLampValues** `[protected]`

4.9.4.6 bool **Lamps::running** `[protected]`

4.9.4.7 double **Lamps::updateRate** `[protected]`

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

- src/lamps.h
- src/lamps.cpp

4.10 Lightprobe Class Reference

Our light probe model.

`#include <lightprobe.h>`

Data Structures

- struct params

Configuration of the light probe.

- struct samplingParams

  Configures sampling.

Public Member Functions

- Lightprobe (Source $*$, params c, samplingParams sampling)
- Lightprobe (Source $*$, string configFile, samplingParams sampling)
- Lightprobe (Source $*$, params c, samplingParams sampling, directions sampling-Dirs)
- ∼Lightprobe ()
- params getConfig ()
- samplingParams getSamplingConfig ()
- Source $*$ getSource ()
- void acquire ()
- imglib::Image$<$ float $>$ & getImage ()
- bool hasNewData ()
- void setRotationY (double rad)
- void precalculateDirectionPixelData ()
- void precalculateSamplingDirections ()
- void precalculateSamplingStructure ()
- void precalculateSamplingCones ()
- void precalculateSamplingNearestNeighbors ()
- void precalculateSamplingAllPixels ()
- vector$<$ rgb $>$ getImpact ()
- vector$<$ rgb $>$ getImpact (imglib::Image$<$ float $>$ &img)
- Vector3d getDirectionFromPixel (Vector2i pos)
- Vector3d getDirectionFromPixelDebevec (Vector2i pos)

Data Fields

- vector$<$ Vector2i $>$ allPixels
- directions allDirs
- directions samplingDirs
- vector$<$ dirCone $>$ samplingCones
- vector$<$ bool $>$ usedDirections

Private Member Functions

- void init ()

Private Attributes

- params config
- samplingParams samplingConfig
- Source ∗ source
- imglib::Image< float > maskImage
- double planeShift

4.10.1 Detailed Description

Our light probe model.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 Lightprobe::Lightprobe (Source ∗ *s*, params *c*, samplingParams *sampling*)

Set up light probe model from a given model config and sampling config.

4.10.2.2 Lightprobe::Lightprobe (Source ∗ *s*, string *configFile*, samplingParams *sampling*)

Set up light probe model using a given sampling configuration. Loads model config from file.

4.10.2.3 Lightprobe::Lightprobe (Source ∗ *s*, params *c*, samplingParams *sampling*, directions *samplingDirs*)

Set up light probe model using a given sampling configuration and sampling directions. Loads model config from file.

4.10.2.4 Lightprobe::∼Lightprobe ( )

4.10.3 Member Function Documentation

4.10.3.1 void Lightprobe::acquire ( )

4.10.3.2 Lightprobe::params Lightprobe::getConfig ( )

4.10.3.3 Vector3d Lightprobe::getDirectionFromPixel (Vector2i *pos*)

Our light probe model. Calculates the reflected light direction from pixel coordinates.

4.10.3.4 Vector3d Lightprobe::getDirectionFromPixelDebevec (Vector2i *pos*)

Light probe model that use the Debevec parametrisation.

4.10.3.5 imglib::Image< float > & Lightprobe::getImage ( )

4.10.3.6 vector$<$ rgb $>$ Lightprobe::getImpact ( )

Calculates the impact of a lamp: Acquires an image from the source, performs down-sampling and returns the sampled values.

4.10.3.7 vector$<$ rgb $>$ Lightprobe::getImpact (imglib::Image$<$ float $>$ & *img*)

Samples a light probe image.

4.10.3.8 Lightprobe::samplingParams Lightprobe::getSamplingConfig ( )

4.10.3.9 Source $*$ Lightprobe::getSource ( )

4.10.3.10 bool Lightprobe::hasNewData ( )

4.10.3.11 void Lightprobe::init ( ) [private]

Initializes the light probe model.

4.10.3.12 void Lightprobe::precalculateDirectionPixelData ( )

Precalculates the direction of reflected light for all available, unmasked pixels within the sampling range.

4.10.3.13 void Lightprobe::precalculateSamplingAllPixels ( )

Generates sampling structure for all-pixel sampling (every pixel becomes one direction).

4.10.3.14 void Lightprobe::precalculateSamplingCones ( )

Generate sampling data structure for Gaussian sampling.

4.10.3.15 void Lightprobe::precalculateSamplingDirections ( )

Precalculates the sampling directions.

4.10.3.16 void Lightprobe::precalculateSamplingNearestNeighbors ( )

Generates sampling datastructure for nearest-neighbor sampling.

4.10.3.17 void Lightprobe::precalculateSamplingStructure ( )

Generates the sampling data structure.

4.10.3.18 void Lightprobe::setRotationY (double *rad*)

Set new value for rotation around y axis (on planar plane) and recalculate sampling datastructures

4.10.4 Field Documentation

4.10.4.1 directions Lightprobe::allDirs

4.10.4.2 vector<Vector2i> Lightprobe::allPixels

4.10.4.3 params Lightprobe::config [private]

4.10.4.4 imglib::Image<float> Lightprobe::maskImage [private]

4.10.4.5 double Lightprobe::planeShift [private]

4.10.4.6 vector<dirCone> Lightprobe::samplingCones

4.10.4.7 samplingParams Lightprobe::samplingConfig [private]

4.10.4.8 directions Lightprobe::samplingDirs

4.10.4.9 Source∗ Lightprobe::source [private]

4.10.4.10 vector<bool> Lightprobe::usedDirections

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

- src/lightprobe.h
- src/lightprobe.cpp

4.11 Lightprobe::params Struct Reference

Configuration of the light probe.

`#include <lightprobe.h>`

Public Member Functions

- params ()
- void load (string file)
- void save (string file)

Data Fields

- double camDistance
- double sphereRadius
- circle sphereCircle
- double gamma
- rgb whitepoint
- double exposure
- Vector3d rotation
- double horizonAngle
- string responseCurve

- string maskFile
- int type

4.11.1 Detailed Description

Configuration of the light probe.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 Lightprobe::params::params ( ) [inline]

4.11.3 Member Function Documentation

4.11.3.1 void Lightprobe::params::load (string *file*)

Loads model config from a file.

4.11.3.2 void Lightprobe::params::save (string *file*)

Saves model config to a file.

4.11.4 Field Documentation

4.11.4.1 double Lightprobe::params::camDistance

4.11.4.2 double Lightprobe::params::exposure

4.11.4.3 double Lightprobe::params::gamma

4.11.4.4 double Lightprobe::params::horizonAngle

4.11.4.5 string Lightprobe::params::maskFile

4.11.4.6 string Lightprobe::params::responseCurve

4.11.4.7 Vector3d Lightprobe::params::rotation

4.11.4.8 circle Lightprobe::params::sphereCircle

4.11.4.9 double Lightprobe::params::sphereRadius

4.11.4.10 int Lightprobe::params::type

4.11.4.11 rgb Lightprobe::params::whitepoint

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

- src/lightprobe.h
- src/lightprobe.cpp

4.12 Lightprobe::samplingParams Struct Reference

Configures sampling.

```
#include <lightprobe.h>
```

Public Types

- enum { UNIFORM_OLD, UNIFORM, FROM_FILE, ALLPIXELS }
- enum { NEIGHBOR, GAUSS, NONE }

Public Member Functions

- samplingParams ()

Data Fields

- enum Lightprobe::samplingParams:: { ... } samplingMode
- int numSamplesH
- int numSamplesA
- int numSamples
- string filename
- enum Lightprobe::samplingParams:: { ... } kernelMode
- double coneSize
- double coneSigma
- int minConeSize

4.12.1 Detailed Description

Configures sampling.

4.12.2 Member Enumeration Documentation

4.12.2.1 anonymous enum

Enumerator:

UNIFORM_OLD

UNIFORM

FROM_FILE

ALLPIXELS

4.12.2.2 anonymous enum

Enumerator:

NEIGHBOR

GAUSS

NONE

4.12.3 Constructor & Destructor Documentation

4.12.3.1 Lightprobe::samplingParams::samplingParams ( ) `[inline]`

4.12.4 Field Documentation

4.12.4.1 double Lightprobe::samplingParams::coneSigma

4.12.4.2 double Lightprobe::samplingParams::coneSize

4.12.4.3 string Lightprobe::samplingParams::filename

4.12.4.4 enum { ... } Lightprobe::samplingParams::kernelMode

4.12.4.5 int Lightprobe::samplingParams::minConeSize

4.12.4.6 int Lightprobe::samplingParams::numSamples

4.12.4.7 int Lightprobe::samplingParams::numSamplesA

4.12.4.8 int Lightprobe::samplingParams::numSamplesH

4.12.4.9 enum { ... } Lightprobe::samplingParams::samplingMode

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

- src/lightprobe.h

4.13 Log Class Reference

A simple logger.

`#include <utils.h>`

Public Member Functions

- Log ()
- virtual ∼Log ()
- ostringstream & log (int lvl)
- ostringstream & msg ()
- ostringstream & err ()

Private Attributes

- ostringstream ss
- int level

4.13.1 Detailed Description

A simple logger.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 Log::Log ( ) `[inline]`

4.13.2.2 Log::∼Log ( ) `[virtual]`

4.13.3 Member Function Documentation

4.13.3.1 std::ostringstream & Log::err ( )

4.13.3.2 std::ostringstream & Log::log (int *lvl*)

4.13.3.3 std::ostringstream & Log::msg ( )

4.13.4 Field Documentation

4.13.4.1 int Log::level `[private]`

4.13.4.2 ostringstream Log::ss `[private]`

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

- src/utils.h
- src/utils.cpp

4.14 MaxExposure Class Reference

Adjusts the Exposure of a UVC webcam.

```
#include <maxexposure.h>
```

Public Member Functions

- MaxExposure (Source ∗s)
- ∼MaxExposure ()
- void run ()
- int getExposure ()
- void setExposure (int)

Private Attributes

- Source ∗ source
- int exposure

Static Private Attributes

- static const string videoDevice = "/dev/video0"

4.14.1 Detailed Description

Adjusts the Exposure of a UVC webcam.

Author

Manuel Jerger <`nom@nomnom.de`>

For maximizing the exposure of an UVC controlled webcam.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 MaxExposure::MaxExposure (Source ∗ *s*)

4.14.2.2 MaxExposure::∼MaxExposure ( ) `[inline]`

4.14.3 Member Function Documentation

4.14.3.1 int MaxExposure::getExposure ( ) `[inline]`

4.14.3.2 void MaxExposure::run ( )

4.14.3.3 void MaxExposure::setExposure (int *exp*)

Set exposure on uvc video device.

4.14.4 Field Documentation

4.14.4.1 int MaxExposure::exposure `[private]`

4.14.4.2 Source∗ MaxExposure::source `[private]`

4.14.4.3 const string MaxExposure::videoDevice = ”/dev/video0” `[static, private]`

Author

Manuel Jerger <nom@nomnom.de>

For maximizing the exposure of an UVC controlled webcam.

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

- src/maxexposure.h
- src/maxexposure.cpp

4.15 point Struct Reference

A point.

`#include <utils.h>`

Public Member Functions

- point (double x, double y)
- point ()

Data Fields

- double x
- double y

4.15.1 Detailed Description

A point.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 point::point (double *x*, double *y*) `[inline]`

4.15.2.2 point::point ( ) `[inline]`

4.15.3 Field Documentation

4.15.3.1 double point::x

4.15.3.2 double point::y

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

- src/utils.h

4.16 rect Struct Reference

A rectangle.

`#include <utils.h>`

Public Member Functions

- rect (double x, double y, double w, double h)
- rect ()
- bool isValid ()

Data Fields

- double x
- double y
- double w
- double h

4.16.1 Detailed Description

A rectangle.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 rect::rect (double *x*, double *y*, double *w*, double *h*) `[inline]`

4.16.2.2 rect::rect ( ) `[inline]`

4.16.3 Member Function Documentation

4.16.3.1 bool rect::isValid ( ) `[inline]`

4.16.4 Field Documentation

4.16.4.1 double rect::h

4.16.4.2 double rect::w

4.16.4.3 double rect::x

4.16.4.4 double rect::y

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

- src/utils.h

4.17 rgb Struct Reference

RGB color.

```
#include <utils.h>
```

Public Member Functions

- rgb ()
- rgb (double r, double g, double b)
- rgb (vector< double > vec)
- vector< double > getVec ()

Data Fields

- double r
- double g
- double b

4.17.1 Detailed Description

RGB color.

4.17.2 Constructor & Destructor Documentation

4.17.2.1 rgb::rgb ( ) `[inline]`

4.17.2.2 rgb::rgb (double *r*, double *g*, double *b*) `[inline]`

4.17.2.3 rgb::rgb (vector< double > *vec*) `[inline]`

4.17.3 Member Function Documentation

4.17.3.1 vector<double> rgb::getVec ( ) `[inline]`

4.17.4 Field Documentation

4.17.4.1 double rgb::b

4.17.4.2 double rgb::g

4.17.4.3 double rgb::r

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

- src/utils.h

4.18 Sandbox Class Reference

A sandbox for experiments.

`#include <sandbox.h>`

Public Member Functions

- Sandbox (Source ∗s, Lightprobe ∗p, Lamps ∗l)
- ∼Sandbox ()
- void run ()

Private Attributes

- Source ∗ source
- Lightprobe ∗ probe
- Lamps ∗ lamps

4.18.1 Detailed Description

A sandbox for experiments.

Author

Manuel Jerger <`nom@nomnom.de`>

Sandbox for experiments.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 Sandbox::Sandbox (Source ∗ *s*, Lightprobe ∗ *p*, Lamps ∗ *l*) `[inline]`

4.18.2.2 Sandbox::∼Sandbox ( ) `[inline]`

4.18.3 Member Function Documentation

4.18.3.1 void Sandbox::run ( )

Author

Manuel Jerger <`nom@nomnom.de`>

Sandbox for experiments.

4.18.4 Field Documentation

4.18.4.1 Lamps$*$ Sandbox::lamps `[private]`

4.18.4.2 Lightprobe$*$ Sandbox::probe `[private]`

4.18.4.3 Source$*$ Sandbox::source `[private]`

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

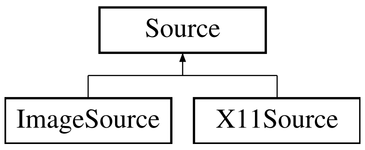
- src/sandbox.h
- src/sandbox.cpp

4.19 Source Class Reference

Acquires and linearizes images.

`#include <source.h>`

Inheritance diagram for Source:



Public Member Functions

- Source ()
- ∼Source ()
- int getWidth ()
- int getHeight ()
- imglib::Image< float > & getImage ()
- void setResponseCurve (string)
- void setWhitepoint (rgb wp)
- void setExposure (double exp)
- virtual bool hasNewData ()=0
- virtual void acquire ()=0
- void start ()
- void stop ()

Protected Types

- enum { RESPONSE_LINEAR, RESPONSE_SRGB, RESPONSE_FILE }

Protected Member Functions

- void loadResponseCurve (string filename)
- void normalizeResponse ()
- double linearize (double value, int channel)
- imglib::Image< float > & linearize (imglib::Image< float > &)

Static Protected Member Functions

- static void * start_thread (void *ptr)

Protected Attributes

- bool running
- bool locked
- imglib::Image< float > imageBuffer
- imglib::Image< float > imageCopy
- int width
- int height
- double exposure
- rgb whitepoint
- int responseSize
- double * responseCurve [3]
- enum Source:: { ... } responseType
- double updateRate

4.19.1 Detailed Description

Acquires and linearizes images.

Author

Manuel Jerger <nom@nomnom.de>

The Source class acquires, linearizes and returns images. Supports threading.

4.19.2 Member Enumeration Documentation

4.19.2.1 anonymous enum `[protected]`

Enumerator:

RESPONSE_LINEAR

RESPONSE_SRGB

RESPONSE_FILE

4.19.3 Constructor & Destructor Documentation

4.19.3.1 Source::Source ( )

Author

Manuel Jerger <nom@nomnom.de>

The Source class acquires, linearizes and returns images. Supports threading.

4.19.3.2 Source::∼Source ( )

4.19.4 Member Function Documentation

4.19.4.1 virtual void Source::acquire ( ) `[pure virtual]`

Implemented in X11Source, and ImageSource.

4.19.4.2 int Source::getHeight ( ) `[inline]`

4.19.4.3 imglib::Image< float > & Source::getImage ( )

4.19.4.4 int Source::getWidth ( ) `[inline]`

4.19.4.5 virtual bool Source::hasNewData ( ) `[pure virtual]`

Implemented in X11Source, and ImageSource.

4.19.4.6 double Source::linearize (double *value,* int *channel*) `[protected]`

Linearizes a single value using the supplied response curve and maps the white point.

4.19.4.7 imglib::Image< float > & Source::linearize (imglib::Image< float > & *A*) `[protected]`

Linearizes an image using the supplied response curve and maps the white point.

4.19.4.8 void Source::loadResponseCurve (string *filename*) `[protected]`

loads an three-channel response curve (either .m format created with hdrcalibrate, or a white-space separated three-column list)

4.19.4.9 void Source::normalizeResponse ( ) `[protected]`

Normalizes the response curve so the image values fits in the range (0:1). The largest value of the three channels of the response curve is mapped to 1.0. The largest value at index 0 is mapped to zero, so we have no positive offset. All three channels are scaled with the same value to preserve the relative relation.

4.19.4.10 void Source::setExposure (double *exp*)

4.19.4.11 void Source::setResponseCurve (string *reponseStr*)

4.19.4.12 void Source::setWhitepoint (rgb *wp*)

4.19.4.13 void Source::start ( )

4.19.4.14 void ∗ Source::start_thread (void ∗ *ptr*) [static, protected]

4.19.4.15 void Source::stop ( )

4.19.5 Field Documentation

4.19.5.1 double Source::exposure [protected]

4.19.5.2 int Source::height [protected]

4.19.5.3 imglib::Image<float> Source::imageBuffer [protected]

4.19.5.4 imglib::Image<float> Source::imageCopy [protected]

4.19.5.5 bool Source::locked [protected]

4.19.5.6 double∗ Source::responseCurve[3] [protected]

4.19.5.7 int Source::responseSize [protected]

4.19.5.8 enum { ... } Source::responseType [protected]

4.19.5.9 bool Source::running [protected]

4.19.5.10 double Source::updateRate [protected]

4.19.5.11 rgb Source::whitepoint [protected]

4.19.5.12 int Source::width [protected]

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

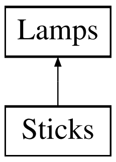
- src/source.h
- src/source.cpp

4.20 Sticks Class Reference

Our sticks lighting system.

`#include <sticks.h>`

Inheritance diagram for Sticks:



Data Structures

- struct params

  Configuration of our lighting system.

Public Member Functions

- Sticks (params config)
- Sticks (int segmentSize, string device, double rate)
- ∼Sticks ()
- params getConfig ()
- bool hasRGBLamps ()
- int getNumRGBLamps ()
- void setRGBValue (int lampID, rgb)
- rgb getRGBValue (int lampID)
- void setAllRGB (rgb)
- int getNumSticks ()
- int getStickLength (int lampID)
- void setStickRGBValue (int stickID, int lampID, rgb)
- rgb getStickRGBValue (int stickID, int lampID)
- void setStickChannelValue (int stickID, int stickLampID, double val, int channel)
- void setAllChannel (double val, int channel)
- bool isReady ()
- bool send ()

Private Member Functions

- void init ()
- unsigned char mapMono (double brightness)

Private Attributes

- params config
- int stripLengths [8]
- int maxStripLength
- int realStripLength
- int maxNumSegs
- vector< unsigned char > rawValues [8][3]
- int serialPort

4.20.1 Detailed Description

Our sticks lighting system.

4.20.2 Constructor & Destructor Documentation

4.20.2.1 Sticks::Sticks (params *c*)

Author

Manuel Jerger <nom@nomnom.de>

This class represents our lighting system. Controls eight strips of 120 WS2812 LEDs via the Teensy microcontroller over the serial port. LEDs on a strip are segmented into a equal sized patches.

4.20.2.2 Sticks::Sticks (int *segmentSize,* string *device,* double *rate*)

4.20.2.3 Sticks::∼Sticks ( )

4.20.3 Member Function Documentation

4.20.3.1 Sticks::params Sticks::getConfig ( )

4.20.3.2 int Sticks::getNumRGBLamps ( )

4.20.3.3 int Sticks::getNumSticks ( )

4.20.3.4 rgb Sticks::getRGBValue (int *lampID*)

4.20.3.5 int Sticks::getStickLength (int *lampID*)

4.20.3.6 rgb Sticks::getStickRGBValue (int *stickID,* int *lampID*)

4.20.3.7 bool Sticks::hasRGBLamps ( )

4.20.3.8 void Sticks::init ( ) `[private]`

4.20.3.9 bool Sticks::isReady ( ) `[inline, virtual]`

Implements Lamps.

4.20.3.10 unsigned char Sticks::mapMono (double *brightness*) `[private]`

4.20.3.11 bool Sticks::send ( ) `[virtual]`

Implements Lamps.

4.20.3.12 void Sticks::setAllChannel (double *val,* int *channel*)

4.20.3.13 void Sticks::setAllRGB (rgb *color*)

4.20.3.14 void **Sticks::setRGBValue** (int *lampID,* rgb *color*)

4.20.3.15 void **Sticks::setStickChannelValue** (int *stickID,* int *stickLampID,* double *val,* int *channel*)

4.20.3.16 void **Sticks::setStickRGBValue** (int *stickID,* int *lampID,* rgb *color*)

4.20.4 Field Documentation

4.20.4.1 params **Sticks::config** `[private]`

4.20.4.2 int **Sticks::maxNumSegs** `[private]`

4.20.4.3 int **Sticks::maxStripLength** `[private]`

4.20.4.4 vector<unsigned char> **Sticks::rawValues[8][3]** `[private]`

4.20.4.5 int **Sticks::realStripLength** `[private]`

4.20.4.6 int **Sticks::serialPort** `[private]`

4.20.4.7 int **Sticks::stripLengths[8]** `[private]`

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

- src/sticks.h
- src/sticks.cpp

4.21 Sticks::params Struct Reference

Configuration of our lighting system.

`#include <sticks.h>`

Public Member Functions

- params ()
- void load (string file)

Data Fields

- int segmentSize
- int numSticks
- int stickSize
- double fadeSpeed
- string serialDevice
- double updateRate

4.21.1 Detailed Description

Configuration of our lighting system.

4.21.2 Constructor & Destructor Documentation

4.21.2.1 Sticks::params::params ( ) `[inline]`

4.21.3 Member Function Documentation

4.21.3.1 void Sticks::params::load (string *file*)

4.21.4 Field Documentation

4.21.4.1 double Sticks::params::fadeSpeed

4.21.4.2 int Sticks::params::numSticks

4.21.4.3 int Sticks::params::segmentSize

4.21.4.4 string Sticks::params::serialDevice

4.21.4.5 int Sticks::params::stickSize

4.21.4.6 double Sticks::params::updateRate

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

- src/sticks.h
- src/sticks.cpp

4.22 TestLamps Class Reference

Test lamps (for debug).

`#include <testlamps.h>`

Public Member Functions

- TestLamps (Lamps ∗l, Source ∗s)
- ∼TestLamps ()
- void run ()

Private Attributes

- Lamps ∗ lamps
- Source ∗ source

4.22.1 Detailed Description

Test lamps (for debug).

Author

Manuel Jerger <nom@nomnom.de>

Old class for testing and debugging the lamps.

4.22.2 Constructor & Destructor Documentation

4.22.2.1 TestLamps::TestLamps (Lamps ∗ *l*, Source ∗ *s*) [inline]

4.22.2.2 TestLamps::∼TestLamps ( )

Author

Manuel Jerger <nom@nomnom.de>

Old class for testing and debugging the lamps.

4.22.3 Member Function Documentation

4.22.3.1 void TestLamps::run ( )

4.22.4 Field Documentation

4.22.4.1 Lamps∗ TestLamps::lamps [private]

4.22.4.2 Source∗ TestLamps::source [private]

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

- src/testlamps.h
- src/testlamps.cpp

4.23 TestProbe Class Reference

Test light probe (for debug).

`#include <testprobe.h>`

Public Member Functions

- TestProbe (Lightprobe ∗p)
- ∼TestProbe ()
- void run ()

Private Attributes

- Lightprobe ∗ probe

4.23.1 Detailed Description

Test light probe (for debug).

Author

Manuel Jerger <`nom@nomnom.de`>

Old class for testing and debugging the probe.

4.23.2 Constructor & Destructor Documentation

4.23.2.1 TestProbe::TestProbe (Lightprobe ∗ *p*) `[inline]`

4.23.2.2 TestProbe::∼TestProbe ( )

Author

Manuel Jerger <`nom@nomnom.de`>

Old class for testing and debugging the probe.

4.23.3 Member Function Documentation

4.23.3.1 void TestProbe::run ( )

4.23.4 Field Documentation

4.23.4.1 Lightprobe∗ TestProbe::probe `[private]`

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

- src/testprobe.h
- src/testprobe.cpp

4.24 Transfer Class Reference

The Ambient Light Transfer loop.

`#include <transfer.h>`

Data Structures

- class CostSimple

  Faster CostFunction for ceres.
- struct params

  Configuration.
- struct Residual

  CostFunction for ceres.

Public Member Functions

- Transfer (Lightprobe ∗p, Lamps ∗l, params c)
- ∼Transfer ()
- params getConfig ()
- void run ()
- void repaint ()
- void exp_plot_kernel (vector< dirCone > cones)

Private Member Functions

- void createResults (int)
- bool loadImpactData ()
- void prepareDataCeres ()
- void runCeres ()
- void prepareDataCVXOPT ()
- void runCVXOPT ()
- bool toggleByKey (bool var, int key)

Private Attributes

- params config
- Lightprobe ∗ probe
- Lightprobe ∗ caliProbe
- Lamps ∗ lamps
- Gui ∗ gui
- int numLamps
- int numDirs
- int numSamples
- int width
- int height
- int width_cali
- int height_cali
- vector< imglib::Image< float > > impactImages
- vector< rgb > maximumImpacts
- vector< vector< rgb > > lightImpacts

- vector< int > samplingDirectionsNearestPixel
- double averageBrightness
- imglib::Image< float > targetImage
- vector< rgb > targetImpact
- bool scaleImpact
- bool lowPrecision
- bool resetWeights
- double targetScale
- double * weights
- double * targetData
- double * impactData
- PyObject * qpsolver
- PyObject * qpsolverArgs
- PyObject * qp_c
- PyObject * qp_Q
- PyObject * qp_A
- PyObject * qp_b
- bool drawTarget
- int drawSamplingCones
- bool drawPseudoResult
- bool drawPseudoResultCones
- bool drawDifference
- bool doAutoAdjust
- double drawScalingFactor
- int keyPressFlag

4.24.1 Detailed Description

The Ambient Light Transfer loop.

4.24.2 Constructor & Destructor Documentation

4.24.2.1 Transfer::Transfer (Lightprobe * *p*, Lamps * *l*, params *c*)

Author

Manuel Jerger <nom@nomnom.de>

Implements the Ambient Light Transfer loop.

4.24.2.2 Transfer::∼Transfer ( )

4.24.3 Member Function Documentation

4.24.3.1 void Transfer::createResults (int *iter*) [private]

Dump image results if config.output specifies a directory/prefix

4.24.3.2 void Transfer::exp_plot_kernel (vector< dirCone > *cones*)

Experimental: was used to analyze the reconstruction quality of the Gaussian sampling. Reconstructs a white image using the Sampling datastructure and dumps the image as well as values of horizontal slices.

4.24.3.3 params Transfer::getConfig ( )

4.24.3.4 bool Transfer::loadImpactData ( ) `[private]`

Loads the calibration data from config.dataDir and performs the sampling.

4.24.3.5 void Transfer::prepareDataCeres ( ) `[private]`

Sets up Ceres as optimizer.

4.24.3.6 void Transfer::prepareDataCVXOPT ( ) `[private]`

Prepares the CVXOPT optimizer. Creates all matrices and vectors.

4.24.3.7 void Transfer::repaint ( )

Repaints the GUI.

4.24.3.8 void Transfer::run ( )

Starts the Ambient Light Transfer

4.24.3.9 void Transfer::runCeres ( ) `[private]`

Starts the optimization.

4.24.3.10 void Transfer::runCVXOPT ( ) `[private]`

Starts the optimization

4.24.3.11 bool Transfer::toggleByKey (bool *var,* int *key*) `[private]`

Returns the inverted value of a boolean iff a key is pressed

4.24.4 Field Documentation

4.24.4.1 double Transfer::averageBrightness `[private]`

4.24.4.2 Lightprobe∗ Transfer::caliProbe `[private]`

4.24.4.3 params Transfer::config `[private]`

4.24.4.4 bool Transfer::doAutoAdjust `[private]`

4.24.4.5 bool Transfer::drawDifference `[private]`

4.24.4.6 bool **Transfer::drawPseudoResult** `[private]`

4.24.4.7 bool **Transfer::drawPseudoResultCones** `[private]`

4.24.4.8 int **Transfer::drawSamplingCones** `[private]`

4.24.4.9 double **Transfer::drawScalingFactor** `[private]`

4.24.4.10 bool **Transfer::drawTarget** `[private]`

4.24.4.11 **Gui**$*$ **Transfer::gui** `[private]`

4.24.4.12 int **Transfer::height** `[private]`

4.24.4.13 int **Transfer::height_cali** `[private]`

4.24.4.14 double$*$ **Transfer::impactData** `[private]`

4.24.4.15 vector$<$imglib::Image$<$float$>$ $>$ **Transfer::impactImages** `[private]`

4.24.4.16 int **Transfer::keyPressFlag** `[private]`

4.24.4.17 **Lamps**$*$ **Transfer::lamps** `[private]`

4.24.4.18 vector$<$vector$<$**rgb**$>$ $>$ **Transfer::lightImpacts** `[private]`

4.24.4.19 bool **Transfer::lowPrecision** `[private]`

4.24.4.20 vector$<$**rgb**$>$ **Transfer::maximumImpacts** `[private]`

4.24.4.21 int **Transfer::numDirs** `[private]`

4.24.4.22 int **Transfer::numLamps** `[private]`

4.24.4.23 int **Transfer::numSamples** `[private]`

4.24.4.24 **Lightprobe**$*$ **Transfer::probe** `[private]`

4.24.4.25 PyObject$*$ **Transfer::qp_A** `[private]`

4.24.4.26 PyObject$*$ **Transfer::qp_b** `[private]`

4.24.4.27 PyObject$*$ **Transfer::qp_c** `[private]`

4.24.4.28 PyObject$*$ **Transfer::qp_Q** `[private]`

4.24.4.29 PyObject$*$ **Transfer::qpsolver** `[private]`

4.24.4.30 PyObject$*$ **Transfer::qpsolverArgs** `[private]`

4.24.4.31 bool **Transfer::resetWeights** `[private]`

4.24.4.32 vector<int> **Transfer::samplingDirectionsNearestPixel** `[private]`

4.24.4.33 bool **Transfer::scaleImpact** `[private]`

4.24.4.34 double∗ **Transfer::targetData** `[private]`

4.24.4.35 imglib::Image<float> **Transfer::targetImage** `[private]`

4.24.4.36 vector<rgb> **Transfer::targetImpact** `[private]`

4.24.4.37 double **Transfer::targetScale** `[private]`

4.24.4.38 double∗ **Transfer::weights** `[private]`

4.24.4.39 int **Transfer::width** `[private]`

4.24.4.40 int **Transfer::width_cali** `[private]`

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

- src/transfer.h
- src/transfer.cpp

4.25 Transfer::CostSimple Class Reference

Faster CostFunction for ceres.

Public Member Functions

- ∼CostSimple ()
- virtual bool Evaluate (double const ∗const ∗parameters, double ∗residuals, double ∗∗jacobians) const
- CostSimple (double target, double ∗samples)

Private Attributes

- double target
- double ∗ samples

4.25.1 Detailed Description

Faster CostFunction for ceres.

4.25.2 Constructor & Destructor Documentation

4.25.2.1 **Transfer::CostSimple::∼CostSimple ( )** `[inline]`

4.25.2.2 Transfer::CostSimple::CostSimple (double *target*, double ∗ *samples*) `[inline]`

4.25.3 Member Function Documentation

4.25.3.1 virtual bool Transfer::CostSimple::Evaluate (double const ∗const ∗ *parameters*, double ∗ *residuals*, double ∗∗ *jacobians*) const `[inline, virtual]`

4.25.4 Field Documentation

4.25.4.1 double∗ Transfer::CostSimple::samples `[private]`

4.25.4.2 double Transfer::CostSimple::target `[private]`

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

- src/transfer.h

4.26 Transfer::params Struct Reference

Configuration.

`#include <transfer.h>`

Public Types

- enum { OPT, SAMPLER }

Public Member Functions

- params ()

Data Fields

- enum Transfer::params:: { ... } algorithm
- double rate
- string dataDir
- int mode
- string output
- double targetScale
- double rampScaleFrom
- double rampScaleTo
- double rampScaleSteps
- bool dynamicFading
- string exec
- bool useUniform
- int numIterations

- bool useAverageScale
- int numRepeats
- bool driveSeparateColors

4.26.1 Detailed Description

Configuration.

4.26.2 Member Enumeration Documentation

4.26.2.1 anonymous enum

Enumerator:

OPT

SAMPLER

4.26.3 Constructor & Destructor Documentation

4.26.3.1 Transfer::params::params ( ) [inline]

4.26.4 Field Documentation

4.26.4.1 enum { ... } Transfer::params::algorithm

4.26.4.2 string Transfer::params::dataDir

4.26.4.3 bool Transfer::params::driveSeparateColors

4.26.4.4 bool Transfer::params::dynamicFading

4.26.4.5 string Transfer::params::exec

4.26.4.6 int Transfer::params::mode

4.26.4.7 int Transfer::params::numIterations

4.26.4.8 int Transfer::params::numRepeats

4.26.4.9 string Transfer::params::output

4.26.4.10 double Transfer::params::rampScaleFrom

4.26.4.11 double Transfer::params::rampScaleSteps

4.26.4.12 double Transfer::params::rampScaleTo

4.26.4.13 double Transfer::params::rate

4.26.4.14 double Transfer::params::targetScale

4.26.4.15 bool Transfer::params::useAverageScale

4.26.4.16 bool Transfer::params::useUniform

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

- src/transfer.h

4.27 Transfer::Residual Struct Reference

CostFunction for ceres.

Public Member Functions

- Residual (double _target, int _size, double ∗_data)
- template<typename T >
  bool operator() (const T ∗const weights, T ∗residuals) const

Data Fields

- const double target
- const int size
- const double ∗ data

4.27.1 Detailed Description

CostFunction for ceres.

4.27.2 Constructor & Destructor Documentation

4.27.2.1 Transfer::Residual::Residual (double *_target,* int *_size,* double ∗ *_data*) `[inline]`

4.27.3 Member Function Documentation

4.27.3.1 template<typename T > bool Transfer::Residual::operator() (const T ∗const *weights,* T ∗ *residuals*) const `[inline]`

4.27.4 Field Documentation

4.27.4.1 const double∗ Transfer::Residual::data

4.27.4.2 const int Transfer::Residual::size

4.27.4.3 const double Transfer::Residual::target

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

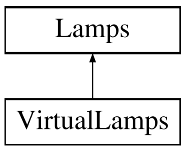
- src/transfer.h

4.28 VirtualLamps Class Reference

Lamps w/o hardware backend.

`#include <virtuallamps.h>`

Inheritance diagram for VirtualLamps:



Public Member Functions

- VirtualLamps (int numLamps)
- ∼VirtualLamps ()
- bool isReady ()
- bool send ()

4.28.1 Detailed Description

Lamps w/o hardware backend.

4.28.2 Constructor & Destructor Documentation

4.28.2.1 VirtualLamps::VirtualLamps (int *numLamps*)

Author

Manuel Jerger <nom@nomnom.de>

This class represents a group of virtual monochrome lamps

4.28.2.2 VirtualLamps::∼VirtualLamps ( ) [inline]

4.28.3 Member Function Documentation

4.28.3.1 bool VirtualLamps::isReady ( ) [virtual]

Implements Lamps.

4.28.3.2 bool VirtualLamps::send ( ) [virtual]

Implements Lamps.

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

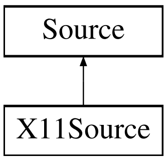
- src/virtuallamps.h
- src/virtuallamps.cpp

4.29 X11Source Class Reference

X11 desktop grabber.

`#include <x11source.h>`

Inheritance diagram for X11Source:



Data Structures

- struct params

  Configuration of X11Source.

Public Member Functions

- X11Source (params c)
- X11Source (string display, rect area, double rate)
- ∼X11Source ()
- params getConfig ()
- void acquire ()
- bool hasNewData ()
- char ∗ getImageRaw ()
- const rect x11SelectAreaOnDesktop (string display)
- const point x11SelectPointOnDesktop (Display ∗disp)

Private Member Functions

- void init ()

Private Attributes

- params config
- string display
- Display ∗ dpy
- XShmSegmentInfo shminfo
- XImage ∗ xImage

4.29.1 Detailed Description

X11 desktop grabber.

Author

Manuel Jerger <`nom@nomnom.de`>

Specialization of the source class that grabs images from the X11 desktop.

4.29.2 Constructor & Destructor Documentation

4.29.2.1 X11Source::X11Source (params *c*)

4.29.2.2 X11Source::X11Source (string *display,* rect *area,* double *rate*)

4.29.2.3 X11Source::∼X11Source ( )

4.29.3 Member Function Documentation

4.29.3.1 void X11Source::acquire ( ) `[virtual]`

Grabs one image from desktop

Implements Source.

4.29.3.2 X11Source::params X11Source::getConfig ( )

4.29.3.3 char ∗ X11Source::getImageRaw ( )

Returns image data straight from X11 shared memory.

4.29.3.4 bool X11Source::hasNewData ( ) `[virtual]`

Implements Source.

4.29.3.5 void X11Source::init ( ) `[private]`

Initialize X11 image grabbing.

4.29.3.6 const rect X11Source::x11SelectAreaOnDesktop (string *display*)

Asks the user to select two points on the desktop that define the top left and bottom right corner of the image area to grab.

4.29.3.7 const point X11Source::x11SelectPointOnDesktop (Display ∗ *disp*)

Asks the user to select a points on the desktop. Uses X11 functions for displaying a cursor and reacting to the click.

4.29.4 Field Documentation

4.29.4.1 params X11Source::config `[private]`

4.29.4.2 string X11Source::display `[private]`

4.29.4.3 Display∗ X11Source::dpy `[private]`

4.29.4.4 XShmSegmentInfo X11Source::shminfo `[private]`

4.29.4.5 XImage∗ X11Source::xImage `[private]`

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

- src/x11source.h
- src/x11source.cpp

4.30 X11Source::params Struct Reference

Configuration of X11Source.

`#include <x11source.h>`

Public Member Functions

- params ()

Data Fields

- string x11Display
- rect captureArea
- double updateRate

4.30.1 Detailed Description

Configuration of X11Source.

4.30.2 Constructor & Destructor Documentation

4.30.2.1 X11Source::params::params ( ) [inline]

4.30.3 Field Documentation

4.30.3.1 rect X11Source::params::captureArea

4.30.3.2 double X11Source::params::updateRate

4.30.3.3 string X11Source::params::x11Display

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

- src/x11source.h

5 File Documentation

5.1 src/alt.cpp File Reference

```
#include "alt.h"
```

Functions

- void parseArgs (int argc, char ∗argv[ ])
- void segfaultHandler (int sig)
- int main (int argc, char ∗argv[ ])

Variables

- int progmode = TRANSFER
- int masterArg = 0
- int verbosity
- Calibrate::params caliConfig
- Transfer::params transferConfig
- Sticks::params sticksConfig
- X11Source::params x11sourceConfig
- ImageSource::params imagesourceConfig
- Lightprobe::params probeConfig
- Lightprobe::samplingParams samplingConfig
- vector< pair< int, double > > setLampsValues
- rgb setLampsRGB
- string setLampsFile
- int numVirtualLamps = 0
- bool useSticks = false

- int sourceType
- bool probeArgs = false
- bool stickArgs = false
- static struct option longOptions []

5.1.1 Function Documentation

5.1.1.1 int main (int *argc,* char ∗ *argv[]*)

Program entry point.

Main() parses the command line arguments and sets up all our classes using the specified configurations. It then runs the specified program mode.

Parameters

argc	Number of arguments.
argv	Arguments

5.1.1.2 void parseArgs (int *argc,* char ∗ *argv[]*)

Parse command line arguments and set the program configuration accordingly.

Parameters

argc	Forwarded argc from main()
argv	Forwarded argv from main()

5.1.1.3 void segfaultHandler (int *sig*)

Add segfault handler and print a backtrace using backtrace() (a feature available in gcc)

Parameters

sig	signal number

5.1.2 Variable Documentation

5.1.2.1 Calibrate::params caliConfig

5.1.2.2 ImageSource::params imagesourceConfig

5.1.2.3 struct option longOptions[] `[static]`

5.1.2.4 int masterArg = 0

5.1.2.5 int numVirtualLamps = 0

5.1.2.6 bool probeArgs = false

5.1.2.7 Lightprobe::params probeConfig

5.1.2.8 int progmode = TRANSFER

5.1.2.9 Lightprobe::samplingParams samplingConfig

5.1.2.10 string setLampsFile

5.1.2.11 rgb setLampsRGB

5.1.2.12 vector<pair<int, double> > setLampsValues

5.1.2.13 int sourceType

5.1.2.14 bool stickArgs = false

5.1.2.15 Sticks::params sticksConfig

5.1.2.16 Transfer::params transferConfig

5.1.2.17 bool useSticks = false

5.1.2.18 int verbosity

Author

Manuel Jerger <nom@nomnom.de>

Utility functions and important datastructures.

5.1.2.19 X11Source::params x11sourceConfig

5.2 src/alt.h File Reference

```
#include "utils.h"  #include "gui.h"  #include "lamps.h"×
#include "lamppool.h"  #include "virtuallamps.h"  #include
"sticks.h"  #include "x11source.h"  #include "lightprobe.-
h"  #include "transfer.h"  #include "calibrate.h"  #include
"testlamps.h" #include "testprobe.h" #include "maxexposure.-
h" #include "sandbox.h" #include <iostream> #include <sstream>×
#include <stdio.h> #include <string.h> #include <vector>×
#include <Eigen/Core>  #include <Eigen/Geometry>  #include
<getopt.h>  #include <X11/Xlib.h>  #include <execinfo.h>×
#include <signal.h>
```

Enumerations

- enum { TRANSFER, TRANSFER_SAMPLER, CAPTURE, CALIBRATE_LAMPS, TESTLAMPS, TESTPROBE, SETLAMPS, MAX_EXPOSURE, SANDBOX, SAMPLE_UNI_OLD, SAMPLE_UNI, SAMPLE_FILE, SAMPLE_ALL, X11SOURCE, IMAGESOURCE, STICKS, VIRTUAL_LAMPS, LIGHTPROBE }

5.2.1 Enumeration Type Documentation

5.2.1.1 anonymous enum

Enumerator:

- ***TRANSFER***
- ***TRANSFER_SAMPLER***
- ***CAPTURE***
- ***CALIBRATE_LAMPS***
- ***TESTLAMPS***
- ***TESTPROBE***
- ***SETLAMPS***
- ***MAX_EXPOSURE***
- ***SANDBOX***
- ***SAMPLE_UNI_OLD***
- ***SAMPLE_UNI***
- ***SAMPLE_FILE***
- ***SAMPLE_ALL***
- ***X11SOURCE***
- ***IMAGESOURCE***
- ***STICKS***
- ***VIRTUAL_LAMPS***
- ***LIGHTPROBE***

5.3 src/calibrate.cpp File Reference

```
#include "calibrate.h"
```

5.4 src/calibrate.h File Reference

```
#include "utils.h" #include "lamps.h" #include "lightprobe.-
h"  #include "gui.h"  #include "ceres/ceres.h"  #include
<glog/logging.h>
```

Data Structures

- class Calibrate

  The Calibration Loop.
- struct Calibrate::params

  Configuration of Calibrate class.

5.5 src/gui.cpp File Reference

`#include "gui.h"`

5.6 src/gui.h File Reference

`#include "utils.h" #include "lamps.h" #include "sticks.h" #include "image.h" #include "source.h" #include <unistd.h> #include "GL/glfw.h" #include <GL/glu.h>`

Data Structures

- class Gui

  The user interface.

5.7 src/imagesource.cpp File Reference

`#include "imagesource.h"`

5.8 src/imagesource.h File Reference

`#include "source.h" #include "image.h" #include <string>`

Data Structures

- class ImageSource

  Source that uses image files.
- struct ImageSource::params

  Configuration of the ImageSource class.

5.9 src/lamppool.cpp File Reference

`#include "lamppool.h"`

5.10 src/lamppool.h File Reference

`#include "lamps.h" #include "utils.h" #include <string> × #include <iostream> #include <vector> #include <pthread.h> #include <time.h>`

Data Structures

- class LampPool

  Groups instances of Lamps.

5.11 src/lamps.cpp File Reference

```
#include "lamps.h"
```

5.12 src/lamps.h File Reference

```
#include "utils.h" #include <vector>
```

Data Structures

- class Lamps

  A monochrome lamp.

5.13 src/lightprobe.cpp File Reference

```
#include "lightprobe.h"
```

5.14 src/lightprobe.h File Reference

```
#include "utils.h" #include "source.h" #include "x11source.-
h" #include "imagesource.h" #include <iostream> #include
<vector> #include <Eigen/Core> #include <Eigen/Geometry> ×
```

Data Structures

- class Lightprobe

  Our light probe model.
- struct Lightprobe::params

  Configuration of the light probe.
- struct Lightprobe::samplingParams

  Configures sampling.

5.15 src/maxexposure.cpp File Reference

```
#include "maxexposure.h"
```

5.16 src/maxexposure.h File Reference

`#include "utils.h" #include "gui.h" #include "source.h"× #include <unistd.h>`

Data Structures

- class MaxExposure

  Adjusts the Exposure of a UVC webcam.

5.17 src/sandbox.cpp File Reference

`#include "sandbox.h"`

5.18 src/sandbox.h File Reference

`#include "utils.h" #include "gui.h" #include "source.-h" #include "lightprobe.h" #include "lamps.h" #include <unistd.h>`

Data Structures

- class Sandbox

  A sandbox for experiments.

5.19 src/source.cpp File Reference

`#include "source.h"`

Defines

- #define FLOOR_NOISE_THRESHOLD (0.000)
- #define HIGHLIGHT_TRESHOLD (0.9)

5.19.1 Define Documentation

5.19.1.1 #define **FLOOR_NOISE_THRESHOLD** (0.000)

5.19.1.2 #define **HIGHLIGHT_TRESHOLD** (0.9)

5.20 src/source.h File Reference

`#include "utils.h"`

Data Structures

- class Source

  Acquires and linearizes images.

5.21 src/sticks.cpp File Reference

`#include "sticks.h"`

5.22 src/sticks.h File Reference

`#include "lamps.h" #include "utils.h" #include <string> #include <iostream> #include <stdlib.h> #include <unistd.h> #include <fcntl.h> #include <errno.h> #include <termios.h> #include <vector> #include <pthread.h> #include <time.h>`

Data Structures

- class Sticks

  Our sticks lighting system.
- struct Sticks::params

  Configuration of our lighting system.

5.23 src/testlamps.cpp File Reference

`#include "testlamps.h"`

5.24 src/testlamps.h File Reference

`#include "lamps.h" #include "source.h" #include "gui.h"`

Data Structures

- class TestLamps

  Test lamps (for debug).

5.25 src/testprobe.cpp File Reference

`#include "testprobe.h"`

5.26 src/testprobe.h File Reference

`#include "lightprobe.h" #include "source.h" #include "gui.-h"`

Data Structures

- class TestProbe

  Test light probe (for debug).

5.27 src/transfer.cpp File Reference

`#include "transfer.h"`

Defines

- #define MIN_WEIGHT_DISTANCE 0.05

5.27.1 Define Documentation

5.27.1.1 #define **MIN_WEIGHT_DISTANCE** 0.05

5.28 src/transfer.h File Reference

```
#include "utils.h"  #include "gui.h"  #include "lamps.h"×
#include "lamppool.h"  #include "virtuallamps.h"  #include
"source.h" #include "imagesource.h" #include "lightprobe.-
h" #include "image.h" #include <vector> #include <time.-
h> #include "GL/glfw.h" #include <GL/glu.h> #include <ceres/ceres.-
h> #include <glog/logging.h> #include "cvxopt.h"
```

Data Structures

- class Transfer

  The Ambient Light Transfer loop.
- struct Transfer::params

  Configuration.
- struct Transfer::Residual

  CostFunction for ceres.
- class Transfer::CostSimple

  Faster CostFunction for ceres.

Defines

- #define PENALTY 100
- #define NUMLAMPS 108

5.28.1 Define Documentation

5.28.1.1 #define **NUMLAMPS** 108

5.28.1.2 #define **PENALTY** 100

5.29 src/utils.cpp File Reference

```
#include "utils.h" #include <getopt.h>
```

Defines

- #define M_SQRT2PI 2.50662827463100050241

Functions

- imglib::Image< float > & imgAdd (imglib::Image< float > &A, imglib::Image< float > &B)
- imglib::Image< float > & imgSub (imglib::Image< float > &A, imglib::Image< float > &B)
- imglib::Image< float > & imgAdd (imglib::Image< float > &A, rgb color)
- imglib::Image< float > & imgMul (imglib::Image< float > &A, float scalar)
- float imgMax (imglib::Image< float > &A)
- float imgMin (imglib::Image< float > &A)
- imglib::Image< float > & imgScale (imglib::Image< float > &A)
- rgb sampleGauss7 (imglib::Image< float > image, int xpos, int ypos)
- double normalDistribution (double sigma, double mu, double x)
- rgb mapGamma (rgb value, double gain, double lambda)
- double mapLinear (double val, double wp, double bp)
- rgb mapLinear (rgb val, rgb wp, rgb bp)
- rgb rgb2srgb (rgb linear)
- double rgb2srgb_component (double value)
- Vector3d rgb2xyY (rgb val)
- rgb srgb2rgb (rgb sRGB)
- double srgb2rgb_component (double value)
- double clamp (double val)
- rgb clamp (rgb val)
- circle getCircumCircle (Vector2d p1, Vector2d p2, Vector2d p3)
- directions sampleHemisphere (int numSamplesH, int numSamplesA)
- directions sampleSphere (int numSamplesH, int numSamplesA, double horizonAngle)

- directions sampleUniform (int numSamples, double horizonAngle)
- directions samplesFromFile (string path, int numVecs, double horizonAngle)
- directions loadVectors3d (string path, int numVecs)
- int findNearestNeighbor (Vector3d vec, directions candidates)
- Vector2d cartesian2spherical (Vector3d cartesian)
- Vector3d spherical2cartesian (Vector2d spherical)
- double angle (Vector2d p1, Vector2d p2)

Variables

- int verbosity = 0
- const int gf7 [7][7]

5.29.1 Define Documentation

5.29.1.1 #define **M_SQRT2PI** 2.50662827463100050241

5.29.2 Function Documentation

5.29.2.1 double **angle** (Vector2d *p1*, Vector2d *p2*)

5.29.2.2 Vector2d **cartesian2spherical** (Vector3d *cartesian*)

5.29.2.3 double **clamp** (double *val*)

5.29.2.4 rgb **clamp** (rgb *val*)

5.29.2.5 int **findNearestNeighbor** (Vector3d *vec*, directions *candidates*)

5.29.2.6 circle **getCircumCircle** (Vector2d *p1*, Vector2d *p2*, Vector2d *p3*)

5.29.2.7 imglib::Image<float>& **imgAdd** (imglib::Image< float > & *A*, imglib::Image< float > & *B*)

5.29.2.8 imglib::Image<float>& **imgAdd** (imglib::Image< float > & *A*, rgb *color*)

5.29.2.9 float **imgMax** (imglib::Image< float > & *A*)

5.29.2.10 float **imgMin** (imglib::Image< float > & *A*)

5.29.2.11 imglib::Image<float>& **imgMul** (imglib::Image< float > & *A*, float *scalar*)

5.29.2.12 imglib::Image<float>& **imgScale** (imglib::Image< float > & *A*)

5.29.2.13 imglib::Image<float>& **imgSub** (imglib::Image< float > & *A*, imglib::Image< float > & *B*)

5.29.2.14 directions **loadVectors3d** (string *path*, int *numVecs*)

5.29.2.15 rgb **mapGamma** (rgb *value*, double *gain*, double *lambda*)

5.29.2.16 double **mapLinear** (double *val*, double *wp*, double *bp*)

5.29.2.17 rgb **mapLinear** (rgb *val*, rgb *wp*, rgb *bp*)

5.29.2.18 double **normalDistribution** (double *sigma*, double *mu*, double *x*)

5.29.2.19 rgb **rgb2srgb** (rgb *linear*)

5.29.2.20 double **rgb2srgb_component** (double *value*)

5.29.2.21 Vector3d **rgb2xyY** (rgb *val*)

5.29.2.22 rgb **sampleGauss7** (imglib::Image< float > *image*, int *xpos*, int *ypos*)

5.29.2.23 directions **sampleHemisphere** (int *numSamplesH*, int *numSamplesA*)

5.29.2.24 directions **samplesFromFile** (string *path*, int *numVecs*, double *horizonAngle*)

5.29.2.25 directions **sampleSphere** (int *numSamplesH*, int *numSamplesA*, double *horizonAngle*)

5.29.2.26 directions **sampleUniform** (int *numSamples*, double *horizonAngle*)

5.29.2.27 Vector3d **spherical2cartesian** (Vector2d *spherical*)

5.29.2.28 rgb **srgb2rgb** (rgb *sRGB*)

5.29.2.29 double **srgb2rgb_component** (double *value*)

5.29.3 Variable Documentation

5.29.3.1 const int **gf7[7][7]**

Initial value:

```
{ { 1, 1, 2, 2, 2, 1, 1 },
                       { 1, 2, 2, 4, 2, 2, 1 },
                       { 2, 2, 4, 8, 4, 2, 2 },
                       { 2, 4, 8,16, 8, 4, 2 },
                       { 2, 2, 4, 8, 4, 2, 2 },
                       { 1, 2, 2, 4, 2, 2, 1 },
                       { 1, 1, 2, 2, 2, 1, 1 } }
```

5.29.3.2 int **verbosity** = 0

Author

Manuel Jerger <nom@nomnom.de>

Utility functions and important datastructures.

5.30 src/utils.h File Reference

`#include "image.h" #include <string> #include <sstream>× #include <iostream> #include <cmath> #include <vector> #include <Eigen/Core> #include <X11/X.h> #include <X11/-Xlib.h> #include <X11/cursorfont.h>`

Data Structures

- class Log

  A simple logger.
- struct dirCone

  Stores the neighborhood of one sampling direction.
- struct rgb

  RGB color.
- struct circle

  A circle.
- struct rect

  A rectangle.
- struct point

  A point.

Typedefs

- typedef vector< Vector3d > directions

Functions

- void err (string msg, bool critical)
- imglib::Image< float > & imgCircularCrop (imglib::Image< float > &imgIn, circle area)
- imglib::Image< float > & imgAdd (imglib::Image< float > &A, imglib::Image< float > &B)
- imglib::Image< float > & imgSub (imglib::Image< float > &A, imglib::Image< float > &B)
- imglib::Image< float > & imgAdd (imglib::Image< float > &A, rgb color)
- imglib::Image< float > & imgMul (imglib::Image< float > &A, float scalar)
- float imgMax (imglib::Image< float > &A)
- float imgMin (imglib::Image< float > &A)
- imglib::Image< float > & imgScale (imglib::Image< float > &A)
- rgb sampleGauss7 (imglib::Image< float > image, int xpos, int ypos)
- double normalDistribution (double sigma, double mu, double x)
- rgb mapGamma (rgb value, double gain, double lambda)
- double mapLinear (double val, double wp, double bp)
- rgb mapLinear (rgb value, rgb whitepoint, rgb blackpoint)

- rgb rgb2srgb (rgb linear)
- double rgb2srgb_component (double value)
- rgb srgb2rgb (rgb sRGB)
- double srgb2rgb_component (double value)
- Vector3d rgb2xyY (rgb val)
- double clamp (double val)
- rgb clamp (rgb val)
- circle getCircumCircle (Vector2d, Vector2d, Vector2d)
- directions sampleHemisphere (int, int)
- directions sampleSphere (int, int, double)
- directions sampleUniform (int numSamples, double horizonAngle)
- directions samplesFromFile (string path, int numVecs, double horizonAngle)
- directions loadVectors3d (string path, int numVecs)
- int findNearestNeighbor (Vector3d vec, directions candidates)
- Vector2d cartesian2spherical (Vector3d cartesian)
- Vector3d spherical2cartesian (Vector2d spherical)
- double angle (Vector2d p1, Vector2d p2)

5.30.1 Typedef Documentation

5.30.1.1 typedef vector<Vector3d> **directions**

5.30.2 Function Documentation

5.30.2.1 double **angle** (Vector2d *p1*, Vector2d *p2*)

5.30.2.2 Vector2d **cartesian2spherical** (Vector3d *cartesian*)

5.30.2.3 double **clamp** (double *val*)

5.30.2.4 rgb **clamp** (rgb *val*)

5.30.2.5 void **err** (string *msg*, bool *critical*)

5.30.2.6 int **findNearestNeighbor** (Vector3d *vec*, directions *candidates*)

5.30.2.7 circle **getCircumCircle** (Vector2d , Vector2d , Vector2d)

5.30.2.8 imglib::Image<float>& **imgAdd** (imglib::Image< float > & *A*, imglib::Image< float > & *B*)

5.30.2.9 imglib::Image<float>& **imgAdd** (imglib::Image< float > & *A*, rgb *color*)

5.30.2.10 imglib::Image<float>& **imgCircularCrop** (imglib::Image< float > & *imgIn*, circle *area*)

5.30.2.11 float **imgMax** (imglib::Image< float > & *A*)

5.30.2.12 float **imgMin** (imglib::Image< float > & *A*)

5.30.2.13 imglib::Image<float>& **imgMul** (imglib::Image< float > & *A*, float *scalar*)

5.30.2.14 imglib::Image<float>& **imgScale** (imglib::Image< float > & *A*)

5.30.2.15 imglib::Image<float>& **imgSub** (imglib::Image< float > & *A*, imglib::Image< float > & *B*)

5.30.2.16 directions **loadVectors3d** (string *path*, int *numVecs*)

5.30.2.17 rgb **mapGamma** (rgb *value*, double *gain*, double *lambda*)

5.30.2.18 double **mapLinear** (double *val*, double *wp*, double *bp*)

5.30.2.19 rgb **mapLinear** (rgb *value*, rgb *whitepoint*, rgb *blackpoint*)

5.30.2.20 double **normalDistribution** (double *sigma*, double *mu*, double *x*)

5.30.2.21 rgb **rgb2srgb** (rgb *linear*)

5.30.2.22 double **rgb2srgb_component** (double *value*)

5.30.2.23 Vector3d **rgb2xyY** (rgb *val*)

5.30.2.24 rgb **sampleGauss7** (imglib::Image< float > *image*, int *xpos*, int *ypos*)

5.30.2.25 directions **sampleHemisphere** (int , int)

5.30.2.26 directions **samplesFromFile** (string *path*, int *numVecs*, double *horizonAngle*)

5.30.2.27 directions **sampleSphere** (int , int , double)

5.30.2.28 directions **sampleUniform** (int *numSamples*, double *horizonAngle*)

5.30.2.29 Vector3d **spherical2cartesian** (Vector2d *spherical*)

5.30.2.30 rgb **srgb2rgb** (rgb *sRGB*)

5.30.2.31 double **srgb2rgb_component** (double *value*)

5.31 src/virtuallamps.cpp File Reference

```
#include "virtuallamps.h"
```

5.32 src/virtuallamps.h File Reference

```
#include "lamps.h" #include "utils.h" #include <string>×
#include <iostream> #include <vector> #include <pthread.-
h> #include <time.h>
```

Data Structures

- class VirtualLamps

  Lamps w/o hardware backend.

5.33 src/x11source.cpp File Reference

```
#include "x11source.h"
```

5.34 src/x11source.h File Reference

```
#include "utils.h"  #include "source.h"  #include "image.-
h" #include <X11/Xlib.h> #include <X11/Xutil.h> #include
<sys/shm.h> #include <X11/extensions/XShm.h>
```

Data Structures

- class X11Source

  X11 desktop grabber.
- struct X11Source::params

  Configuration of X11Source.