GSoC2011SfM

0.1

Generated by Doxygen 1.7.5.1

Sun Aug 21 2011 16:45:52

Contents

Chapter 1

SDK Structure From Motion Documentation

1.1 What's the point?

Structure from motion aims to find both cameras and objects position, orientation and shape.
As this task is complex and highly depends on videos contents, a fast-robust-accurate technic who works with every types of input is still a dream. This API try to give to user an easy way to try differents algorithms for points detection, matching and of course geometry recovery.
So in a long term, you will be able to do, with this API:

- Manage one or several cameras (I mean physical device) in a sequence (stereovision, single camera, multivision...).
- Each camera can be of different type (Fisheye, with/without radial distortion, various intra-parameters...).
- Initialize the different processing blocks according to the data availables
  Camera: Distortion, intra parameters, nothing, . . .
  Field of view: Extern position, points of interest, a known 3D pattern to match, 2D images, . . .
- Compute missing data (intra/extern parameters, 3D points,...)
- Show the points cloud using an interactive visualization

1.2 Example

I made a little video to show current reconstruction progress. This is not really a structure from motion as the cameras are fully parameterized, but it's a start...

You can see it here: http://www.youtube.com/embed/9M4KWgRGNa0. The dependencies of this API are for now:

- Opencv : http://opencv.willowgarage.com
- PCL (Point Cloud Library) : http://pointclouds.org
- The libmv project : http://code.google.com/p/libmv/
- The Eigen library (Needed by PCL and LibMV) : http://eigen.tuxfamily.org/
- The Boost libraries : http://www.boost.org/
- clapack : http://www.netlib.org/clapack/
- lourakis' sba : http://www.ics.forth.gr/~lourakis/sba/

Chapter 2

Class 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

Class Documentation

4.1 OpencvSfM::bundle_datas Struct Reference

```
#include <bundle_related.h>
```

Public Member Functions

- bundle_datas (libmv::vector< libmv::Mat3 > &i, libmv::vector< Eigen::Quaterniond > &r, libmv::vector< libmv::Vec3 > &t, int c, int p, int mp, int n, int m)

Public Attributes

- libmv::vector< libmv::Mat3 > & intraParams

  list of intra parameters of each cameras
- libmv::vector < Eigen::Quaterniond > & rotations

  list of rotations matrix of each cameras
- libmv::vector< libmv::Vec3 > & translations

  list of translation vector of each cameras
- double ∗ points3D

  List of 3d points.
- int cnp

  number of parameters for ONE camera; e.g. 6 for Euclidean cameras
- int pnp

  number of parameters for ONE 3D point; e.g. 3 for Euclidean points
- int mnp

  number of parameters for ONE projected point; e.g. 2 for Euclidean points
- int ncon

  number of points (starting from the 1st) whose parameters should not be modified.

- int mcon

  number of cameras (starting from the 1st) whose parameters should not be modified.

4.1.1 Detailed Description

This structure help lourakis bundle adjustment to find needed information.

Definition at line 16 of file bundle_related.h.

4.1.2 Constructor & Destructor Documentation

4.1.2.1 OpencvSfM::bundle_datas::bundle_datas (libmv::vector< libmv::Mat3 > & *i*, libmv::vector< Eigen::Quaterniond > & *r*, libmv::vector< libmv::Vec3 > & *t*, int *c*, int *p*, int *mp*, int *n*, int *m*) `[inline]`

Construct a bundle helper object.

Parameters

i	list of intra parameters of each cameras
r	list of rotations matrix of each cameras
t	list of translation vector of each cameras
c	number of parameters for ONE camera
p	number of parameters for ONE 3D point
mp	number of parameters for ONE projected point
n	number of points (starting from the 1st) whose parameters should not be modified.
m	number of cameras (starting from the 1st) whose parameters should not be modified.

Definition at line 39 of file bundle_related.h.

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

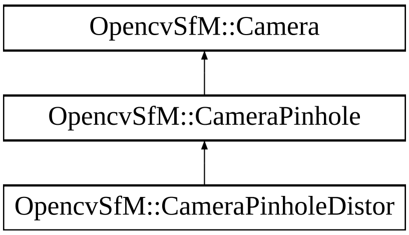
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/bundle_related.h

4.2 OpencvSfM::Camera Class Reference

This class represent the physical device which take the pictures. It is not related to a 3D position which is the role of the PointOfView class. The role of the class is to store only device related informations like intra parameters, radial and tangential distotion. This abstract class is not related to a type of camera (fish eyes...)

```
#include <Camera.h>
```

Inheritance diagram for OpencvSfM::Camera:



Public Member Functions

- virtual std::vector< cv::Vec4d > convertFromImageTo3Dray (std::vector< cv::Vec3d > points)=0
- virtual std::vector< cv::Vec2d > pixelToNormImageCoordinates (std::vector< cv::Vec2d > points) const =0
- virtual std::vector< cv::Vec2d > normImageToPixelCoordinates (std::vector< cv::Vec2d > points) const =0
- virtual cv::Mat getIntraMatrix () const
- virtual double getFocal () const =0
- virtual void write (cv::FileStorage &fs) const =0

Static Public Member Functions

- static cv::Ptr< Camera > read (const cv::FileNode &node)

Protected Member Functions

- Camera ()

Protected Attributes

- std::vector< PointOfView ∗ > pointsOfView_

  vector of the differents positions of the camera.

Friends

- class **PointOfView**

4.2.1 Detailed Description

This class represent the physical device which take the pictures. It is not related to a 3D position which is the role of the PointOfView class. The role of the class is to store only device related informations like intra parameters, radial and tangential distotion. This abstract class is not related to a type of camera (fish eyes...)

This class can be used to store device related informations like intra parameters, radial and tangential distortion. If we use the so-called pinhole camera model, a scene view is formed by projecting 3D points into the image plane using a perspective transformation. Usual notation says that a point [u,v] from an image is related to the point [X,Y,Z] using the following notation :

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

This leads to the following relation between local coordinates and global ones:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = R \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} + t$$

$$x' = x/z$$
$$y' = y/z$$

Additionnal radial and tangeancial distortion are modelized like this:

$$x'' = x' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x' y' + p_2 (r^2 + 2x'^2)$$

$$y'' = y' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + p_1 (r^2 + 2y'^2) + 2p_2 x' y'$$

$$\text{where} \quad r^2 = x'^2 + y'^2$$
$$u = f_x * x'' + c_x$$
$$v = f_y * y'' + c_y$$

radial_dist_ can be used to store k_1 to k_6 tangential_dist_ can be used to store p_1 and p_2

So this class is devoted to the conversion between 2D points from pixel image coordinates and 2D points in normalized image coordinates, or ray projection using intra parameters.

Definition at line 48 of file Camera.h.

4.2.2 Constructor & Destructor Documentation

4.2.2.1 OpencvSfM::Camera::Camera ( ) `[protected]`

As this class is virtual, we can't create a new empty camera...

Definition at line 10 of file Camera.cpp.

4.2.3 Member Function Documentation

4.2.3.1 virtual std::vector<cv::Vec4d> OpencvSfM::Camera::convertFromImageTo3Dray (std::vector< cv::Vec3d > *points*) [pure virtual]

This method can transform points from image to 3D rays (homogeneous coordinates)

Implemented in OpencvSfM::CameraPinholeDistor, and OpencvSfM::CameraPinhole.

4.2.3.2 virtual double OpencvSfM::Camera::getFocal ( ) const [pure virtual]

This method is useful to get the focal from Intrinsic matrix:

Returns

focal lenght

Implemented in OpencvSfM::CameraPinhole.

4.2.3.3 virtual cv::Mat OpencvSfM::Camera::getIntraMatrix ( ) const [inline, virtual]

This method return the intra parameters of the camera

Returns

Matrix K of intra parameters

Reimplemented in OpencvSfM::CameraPinhole.

Definition at line 84 of file Camera.h.

4.2.3.4 virtual std::vector<cv::Vec2d> OpencvSfM::Camera::normImageToPixelCoordinates (std::vector< cv::Vec2d > *points*) const [pure virtual]

This method can convert 2D points from normalized image coordinates to 2D points in pixel image coordinates

Parameters

points	2D points in normalized image homogeneous coordinates.

Returns

2D points in pixel image coordinates.

Implemented in OpencvSfM::CameraPinholeDistor, and OpencvSfM::CameraPinhole.

4.2.3.5 virtual std::vector<cv::Vec2d> OpencvSfM::Camera::pixelToNormImageCoordinates (std::vector< cv::Vec2d > *points*) const [pure virtual]

This method can convert 2D points from pixel image coordinates to 2D points in normalized image coordinates

Parameters

points	2D points in pixel image homogeneous coordinates.

Returns

2D points in normalized image homogeneous coordinates.

Implemented in OpencvSfM::CameraPinholeDistor, and OpencvSfM::CameraPinhole.

4.2.3.6 cv::Ptr< Camera > OpencvSfM::Camera::read (const cv::FileNode & *node*) [static]

Create a new camera from a YAML file.

Parameters

node	Previously opened YAML file node

Reimplemented in OpencvSfM::CameraPinhole, and OpencvSfM::CameraPinholeDistor.

Definition at line 19 of file Camera.cpp.

4.2.3.7 virtual void OpencvSfM::Camera::write (cv::FileStorage & *fs*) const [pure virtual]

Save the camera intra parameters into a YAML file.

Parameters

fs	Previously opened YAML file node

Implemented in OpencvSfM::CameraPinhole, and OpencvSfM::CameraPinholeDistor.

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

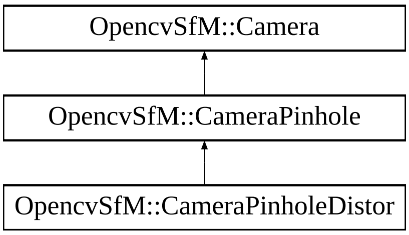
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/Camera.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/Camera.cpp

4.3 OpencvSfM::CameraPinhole Class Reference

This class represent the physical device which take the pictures. It is not related to a 3D position which is the role of the PointOfView class. The role of the class is to store only intra parameters (without radial distortion)

```
#include <CameraPinhole.h>
```

Inheritance diagram for OpencvSfM::CameraPinhole:



Public Member Functions

- CameraPinhole (cv::Mat intra_params=cv::Mat::eye(3, 3, CV_64F), unsigned char wantedEstimation=FOCAL_PARAM|SKEW_PARAM|PRINCIPAL_POINT_PARAM)
- CameraPinhole (const std::vector< std::vector< cv::Point3f > > &objectPoints, const std::vector< std::vector< cv::Point2f > > &imagePoints, cv::Size imageSize, double aspectRatio=1., unsigned char wantedEstimation=FOCAL_PARAM|SKEW_PARAM|PRINCIPAL_POINT_PARAM)
- void updateIntrinsicMatrix (cv::Mat newParams, unsigned char intraValues=FOCAL_PARAM|SKEW_PARAM|PRINCIPAL_POINT_PARAM)
- virtual std::vector< cv::Vec4d > convertFromImageTo3Dray (std::vector< cv::Vec3d > points)
- virtual std::vector< cv::Vec2d > pixelToNormImageCoordinates (std::vector< cv::Vec2d > points) const
- virtual std::vector< cv::Vec2d > normImageToPixelCoordinates (std::vector< cv::Vec2d > points) const
- virtual cv::Mat getIntraMatrix () const
- virtual double getFocal () const
- virtual void write (cv::FileStorage &fs) const

Static Public Member Functions

- static cv::Ptr< Camera > read (const cv::FileNode &node)

Protected Attributes

- cv::Mat intra_params_

store intra parameters(3∗3 matrix). This matrix contains focal informations, principal point coordinates and skew of axis

- cv::Mat inv_intra_params_

  This is the inverse transformation of intra_params_. Used to speed up calculus...

- unsigned char estimation_needed_

4.3.1 Detailed Description

This class represent the physical device which take the pictures. It is not related to a 3D position which is the role of the PointOfView class. The role of the class is to store only intra parameters (without radial distortion)

So this class is devoted to the conversion between 3D points (using camera coordinate) and 2D points (using image coordinate) using the methods convertFromImageTo3Dray or convertFrom3DToImage

Definition at line 24 of file CameraPinhole.h.

4.3.2 Constructor & Destructor Documentation

4.3.2.1 OpencvSfM::CameraPinhole::CameraPinhole (cv::Mat *intra_params* = `cv::Mat::eye( 3, 3, CV_64F )`, unsigned char *wantedEstimation* = `FOCAL_PARAM|SKEW_PARAM|PRINCIPAL_POINT_PARAM`)

Constructor with (or not) intra parameters.

Parameters

intra_params	matrix of intra parameters (3∗3)
wantedEstimation	values which need an estimation

Definition at line 12 of file CameraPinhole.cpp.

Referenced by read().

4.3.2.2 OpencvSfM::CameraPinhole::CameraPinhole (const std::vector< std::vector< cv::Point3f > > & *objectPoints*, const std::vector< std::vector< cv::Point2f > > & *imagePoints*, cv::Size *imageSize*, double *aspectRatio* = `1.`, unsigned char *wantedEstimation* = `FOCAL_PARAM|SKEW_PARAM|PRINCIPAL_POINT_PARAM`)

Constructor where initial camera matrix is computed from the 3D-2D point correspondences. Currently, the function only supports planar calibration patterns, i.e. patterns where each object point has z-coordinate =0.

Parameters

objectPoints	The vector of vectors of the object points. See http://opencv.willowgarage.com/documentation/cpp/calib3d_camera_calibration_and_3d_reconstruction.html#cv-calibratecamera
imagePoints	The vector of vectors of the corresponding image points. See http://opencv.willowgarage.com/documentation/cpp/calib3d_camera_calibration_and_3d_reconstruction.html#cv-calibratecamera
imageSize	The image size in pixels; used to initialize the principal point
aspectRatio	If it is zero or negative, both f_x and f_y are estimated independently. Otherwise $f_x = f_y * aspectRatio$
wantedEstimation	values which need an estimation

Definition at line 21 of file CameraPinhole.cpp.

4.3.3 Member Function Documentation

4.3.3.1 vector< Vec4d > OpencvSfM::CameraPinhole::convertFromImageTo3Dray (std::vector< cv::Vec3d > *points*) [virtual]

This method can transform points from image to 3D rays

Implements OpencvSfM::Camera.

Reimplemented in OpencvSfM::CameraPinholeDistor.

Definition at line 67 of file CameraPinhole.cpp.

4.3.3.2 double OpencvSfM::CameraPinhole::getFocal ( ) const [virtual]

This method retrive the focal from Intrinsic matrix. It's not using pixel reference but using camera reference!

Returns

focal lenght

Implements OpencvSfM::Camera.

Definition at line 122 of file CameraPinhole.cpp.

4.3.3.3 virtual cv::Mat OpencvSfM::CameraPinhole::getIntraMatrix ( ) const [inline, virtual]

This method return the intra parameters of the camera

Returns

Matrix K of intra parameters

Reimplemented from OpencvSfM::Camera.

Definition at line 82 of file CameraPinhole.h.

Referenced by OpencvSfM::CameraPinholeDistor::read().

4.3.3.4 vector< Vec2d > OpencvSfM::CameraPinhole::normImageToPixelCoordinates (std::vector< cv::Vec2d > *points*) const [virtual]

This method can convert 2D points from normalized image coordinates to 2D points in pixel image coordinates

Parameters

points	2D points in normalized image homogeneous coordinates.

Returns

2D points in pixel image coordinates.

Implements OpencvSfM::Camera.

Reimplemented in OpencvSfM::CameraPinholeDistor.

Definition at line 103 of file CameraPinhole.cpp.

4.3.3.5 vector< Vec2d > OpencvSfM::CameraPinhole::pixelToNormImageCoordinates (std::vector< cv::Vec2d > *points*) const [virtual]

This method can convert 2D points from pixel image coordinates to 2D points in normalized image coordinates

Parameters

points	2D points in pixel image homogeneous coordinates.

Returns

2D points in normalized image homogeneous coordinates.

Implements OpencvSfM::Camera.

Reimplemented in OpencvSfM::CameraPinholeDistor.

Definition at line 73 of file CameraPinhole.cpp.

4.3.3.6 cv::Ptr< Camera > OpencvSfM::CameraPinhole::read (const cv::FileNode & *node*) [static]

Create a new camera from a YAML file.

Parameters

node	Previously opened YAML file node

Reimplemented from OpencvSfM::Camera.

Reimplemented in OpencvSfM::CameraPinholeDistor.

Definition at line 154 of file CameraPinhole.cpp.

Referenced by OpencvSfM::PointOfView::read().

4.3.3.7 void OpencvSfM::CameraPinhole::updateIntrinsicMatrix (cv::Mat *newParams*, unsigned char *intraValues* = FOCAL_PARAM|SKEW_PARAM|PRINCIPAL_POINT_PARAM)

this method can be used to update the intra parameters.

Parameters

newParams	matrix of new parameters ($3*3$)
intraValues	values which are useful in matrix

Definition at line 36 of file CameraPinhole.cpp.

4.3.3.8 void OpencvSfM::CameraPinhole::write (cv::FileStorage & *fs*) const [virtual]

Save the camera intra parameters into a YAML file.

Parameters

fs	Previously opened YAML file node

Implements OpencvSfM::Camera.

Reimplemented in OpencvSfM::CameraPinholeDistor.

Definition at line 174 of file CameraPinhole.cpp.

4.3.4 Member Data Documentation

4.3.4.1 unsigned char OpencvSfM::CameraPinhole::estimation_needed_ [protected]

This attribut is used to know what we should estimate... Example: if equal to 0, nothing should be estimated... If equal to 3, focal and skew should be estimated (FOCAL_PARAM + SKEW_PARAM)

Definition at line 34 of file CameraPinhole.h.

Referenced by CameraPinhole(), OpencvSfM::CameraPinholeDistor::updateDistortionParameters(), and write().

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

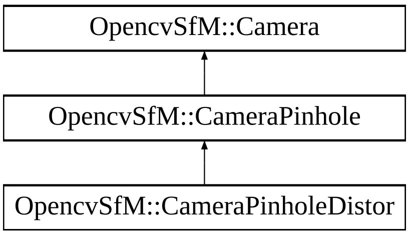
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/CameraPinhole.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/CameraPinhole.cpp

4.4 OpencvSfM::CameraPinholeDistor Class Reference

This class represent the physical device which take the pictures. It is not related to a 3D position which is the role of the PointOfView class. The role of the class is to store intra parameters and radial distortion.

`#include <CameraPinholeDistor.h>`

Inheritance diagram for OpencvSfM::CameraPinholeDistor:



Public Member Functions

- CameraPinholeDistor (cv::Mat intra_params=cv::Mat::eye(3, 3, CV_64F), cv::Vec6d radial_dist=cv::Vec6d(0.0, 0.0, 0.0, 0.0, 0.0, 0.0), unsigned char nbRadialParam=6, cv::Vec2d tangential_dist=cv::Vec2d(0.0, 0.0), unsigned char wantedEstimation=FOCAL_PARAM|SKEW_PARAM|PRINCIPAL_POINT_PARAM|RADIAL_PARAM|TANGEANT_PARAM)
- CameraPinholeDistor (const std::vector< std::vector< cv::Point3f > > &objectPoints, const std::vector< std::vector< cv::Point2f > > &imagePoints, cv::Size imageSize, double aspectRatio=1., cv::Vec6d radial_dist=cv::Vec6d(0.0, 0.0, 0.0,

0.0, 0.0, 0.0), unsigned char nbRadialParam=6, cv::Vec2d tangential_dist=cv::Vec2d(0.0, 0.0), unsigned char wantedEstimation=FOCAL_PARAM|SKEW_PARAM|PRINCIPAL_POINT_PARAM|RADIAL_PARAM|TANGEANT_PARAM)

- void updateDistortionParameters (const cv::Vec6d &radial_dist, unsigned char nbRadialParam, const cv::Vec2d &tangential_dist, unsigned char wantedEstimation=RADIAL_PARAM|TANGEANT_PARAM)
- virtual std::vector< cv::Vec4d > convertFromImageTo3Dray (std::vector< cv::Vec3d > points)
- virtual std::vector< cv::Vec2d > pixelToNormImageCoordinates (std::vector< cv::Vec2d > points) const
- virtual std::vector< cv::Vec2d > normImageToPixelCoordinates (std::vector< cv::Vec2d > points) const
- virtual void write (cv::FileStorage &fs) const

Static Public Member Functions

- static cv::Ptr< Camera > read (const cv::FileNode &node)

Protected Attributes

- cv::Vec< double, 6 > radial_dist_

  used to store radial dist parameters (/fk_1/f to /fk_6/f)
- unsigned char nb_radial_params_

  number of radial dist parameters (0, 2, 3 or 6)
- cv::Vec< double, 2 > tangential_dist_

  used to store tangential dist parameters (/fp_1/f and /fp_2/f)
- unsigned char nb_tangent_params_

  N umbers of tangeancial distorition parameters (0, 1 or 2)
- cv::Mat distortionVector

  vector of distortion coefficients (k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6]]) of 4, 5 or 8 elements

4.4.1 Detailed Description

This class represent the physical device which take the pictures. It is not related to a 3D position which is the role of the PointOfView class. The role of the class is to store intra parameters and radial distortion.

So this class is devoted to the conversion between 3D points (using camera coordinate) and 2D points (using image coordinate) using the methods convertFromImageTo3Dray or convertFrom3DToImage

Definition at line 24 of file CameraPinholeDistor.h.

4.4.2 Constructor & Destructor Documentation

4.4.2.1 OpencvSfM::CameraPinholeDistor::CameraPinholeDistor (cv::Mat *intra_params* = `cv::Mat::eye( 3, 3, CV_64F )`, cv::Vec6d *radial_dist* = `cv::Vec6d( 0.0,0.0,0.0,0.0,0.0,0.0 )`, unsigned char *nbRadialParam* = `6`, cv::Vec2d *tangential_dist* = `cv::Vec2d( 0.0,0.0 )`, unsigned char *wantedEstimation* = `FOCAL_PARAM|SKEW_PARAM|PRINCIPAL_POINT_PARAM|RADIAL_PARAM|TANGEANT_PARAM`)

Constructor with (or not) intra parameters.

Parameters

intra_params	matrix of intra parameters (3∗3)
radial_dist	radial dist parameters (/fk_1/f to /fk_6/f)
nbRadialParam	number of radial dist parameters (0, 2, 3 or 6)
tangential_dist	tangential dist parameters (/fp_1/f and /fp_2/f)
wantedEstimation	values which need an estimation

Definition at line 12 of file CameraPinholeDistor.cpp.

Referenced by read().

4.4.2.2 OpencvSfM::CameraPinholeDistor::CameraPinholeDistor (const std::vector< std::vector< cv::Point3f > > & *objectPoints*, const std::vector< std::vector< cv::Point2f > > & *imagePoints*, cv::Size *imageSize*, double *aspectRatio* = `1.`, cv::Vec6d *radial_dist* = `cv::Vec6d( 0.0,0.0,0.0,0.0,0.0,0.0 )`, unsigned char *nbRadialParam* = `6`, cv::Vec2d *tangential_dist* = `cv::Vec2d( 0.0,0.0 )`, unsigned char *wantedEstimation* = `FOCAL_PARAM|SKEW_PARAM|PRINCIPAL_POINT_PARAM|RADIAL_PARAM|TANGEANT_PARAM`)

Constructor where initial camera matrix is computed from the 3D-2D point correspondences. Currently, the function only supports planar calibration patterns, i.e. patterns where each object point has z-coordinate =0.

Parameters

objectPoints	The vector of vectors of the object points. See http://opencv.willowgarage.com/documentation/cpp/calib3d_camera_calibration_and_3d_reconstruction.html#cv-calibratecamera
imagePoints	The vector of vectors of the corresponding image points. See http://opencv.willowgarage.com/documentation/cpp/calib3d_camera_calibration_and_3d_reconstruction.html#cv-calibratecamera
imageSize	The image size in pixels; used to initialize the principal point
aspectRatio	If it is zero or negative, both f_x and f_y are estimated independently. Otherwise $f_x = f_y * aspectRatio$
radial_dist	radial dist parameters (/fk_1/f to /fk_6/f)
nbRadialParam	number of radial dist parameters (0, 2, 3 or 6)
tangential_dist	tangential dist parameters (/fp_1/f and /fp_2/f)
wantedEstimation	values which need an estimation

Definition at line 20 of file CameraPinholeDistor.cpp.

4.4.3 Member Function Documentation

4.4.3.1 std::vector< cv::Vec4d > OpencvSfM::CameraPinholeDistor::convertFromImageTo3Dray (std::vector< cv::Vec3d > *points*) [virtual]

This method can transform points from image to 3D rays

Reimplemented from OpencvSfM::CameraPinhole.

Definition at line 68 of file CameraPinholeDistor.cpp.

4.4.3.2 vector< Vec2d > OpencvSfM::CameraPinholeDistor::normImageToPixelCoordinates (std::vector< cv::Vec2d > *points*) const [virtual]

This method can convert 2D points from normalized image coordinates to 2D points in pixel image coordinates

Parameters

points	2D points in normalized image homogeneous coordinates.

Returns

2D points in pixel image coordinates.

Reimplemented from OpencvSfM::CameraPinhole.

Definition at line 84 of file CameraPinholeDistor.cpp.

4.4.3.3 vector< Vec2d > OpencvSfM::CameraPinholeDistor::pixelToNormImageCoordinates (std::vector< cv::Vec2d > *points*) const [virtual]

This method can convert 2D points from pixel image coordinates to 2D points in normalized image coordinates

Parameters

points	2D points in pixel image homogeneous coordinates.

Returns

2D points in normalized image homogeneous coordinates.

Reimplemented from OpencvSfM::CameraPinhole.

Definition at line 75 of file CameraPinholeDistor.cpp.

4.4.3.4 cv::Ptr< Camera > OpencvSfM::CameraPinholeDistor::read (const cv::FileNode & *node*) [static]

Create a new camera from a YAML file.

Parameters

node	Previously opened YAML file node

Reimplemented from OpencvSfM::CameraPinhole.

Definition at line 126 of file CameraPinholeDistor.cpp.

Referenced by OpencvSfM::PointOfView::read().

4.4.3.5 void OpencvSfM::CameraPinholeDistor::updateDistortionParameters (const cv::Vec6d & *radial_dist,* unsigned char *nbRadialParam,* const cv::Vec2d & *tangential_dist,* unsigned char *wantedEstimation* = RADIAL_PARAM|TANGEANT_PARAM)

this method can be used to update the intra parameters.

Parameters

radial_dist	values of the new radial distortions parameters

nbRadial-Param	number of radial dist parameters (0, 2, 3 or 6)
tangential_-dist	values of the new tangential distortions parameters
wanted-Estimation	values which need an estimation

Definition at line 35 of file CameraPinholeDistor.cpp.

Referenced by CameraPinholeDistor().

4.4.3.6 void OpencvSfM::CameraPinholeDistor::write (cv::FileStorage & *fs*) const [virtual]

Save the camera intra parameters into a YAML file.

Parameters

fs	Previously opened YAML file node

Reimplemented from OpencvSfM::CameraPinhole.

Definition at line 159 of file CameraPinholeDistor.cpp.

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

- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/CameraPinholeDistor.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/CameraPinholeDistor.cpp

4.5 OpencvSfM::EuclideanEstimator Class Reference

This class perform a projective estimation of the motion. Given points matches and cameras with intra parameters, it tries to find the best cameras positions and 3D points. Does not perform a bundle ajustement!

```
#include <EuclideanEstimator.h>
```

Public Member Functions

- EuclideanEstimator (SequenceAnalyzer &sequence, std::vector< PointOfView > &cameras)
- virtual ∼EuclideanEstimator (void)
- void addNewPointOfView (const PointOfView &camera)
- void computeReconstruction ()
- void bundleAdjustement ()

- void viewEstimation ()
- void initialReconstruction (int image1, int image2)
- bool cameraResection (unsigned int image)

Public Attributes

- std::vector< TrackOfPoints > point_computed_

  list of 3D points computed
- std::vector< bool > camera_computed_

  List of camera computed.

Protected Attributes

- int index_origin

  index of camera set as origin...
- libmv::vector< libmv::Mat3 > intra_params_

  Intra parameters of cameras (don't use them, they are strongly related to cameras_ attribut!
- libmv::vector< libmv::Mat3 > rotations_

  rotations matrix of cameras (don't use them, they are strongly related to cameras_ attribut!
- libmv::vector< libmv::Vec3 > translations_

  translation vectors of cameras (don't use them, they are strongly related to cameras_ attribut!
- std::vector< PointOfView > & cameras_

  List of cameras (intra and extern parameters...)
- SequenceAnalyzer & sequence_

  Object containing all 2D information of this sequence.

4.5.1 Detailed Description

This class perform a projective estimation of the motion. Given points matches and cameras with intra parameters, it tries to find the best cameras positions and 3D points. Does not perform a bundle ajustement!

As this class use a lot of libmv functions, the data members are using libmv structures...

Definition at line 23 of file EuclideanEstimator.h.

4.5.2 Constructor & Destructor Documentation

4.5.2.1 OpencvSfM::EuclideanEstimator::EuclideanEstimator (SequenceAnalyzer & *sequence,* std::vector< PointOfView > & *cameras*)

Construct an euclidean estimator using a sequence of 2D points matches and a list of camera guess (intra parameters should be known!)

Parameters

sequence	Object containing all 2D information of this sequence
cameras	List of cameras (intra (and extern if available) parameters...)

Definition at line 114 of file EuclideanEstimator.cpp.

4.5.2.2 OpencvSfM::EuclideanEstimator::∼EuclideanEstimator (void) [virtual]

Destructor of EuclideanEstimator

Definition at line 127 of file EuclideanEstimator.cpp.

4.5.3 Member Function Documentation

4.5.3.1 void OpencvSfM::EuclideanEstimator::addNewPointOfView (const PointOfView & *camera*)

Add a new camera to the estimator

Parameters

camera	new point of view to add for reconstruction

Definition at line 132 of file EuclideanEstimator.cpp.

Referenced by EuclideanEstimator().

4.5.3.2 void OpencvSfM::EuclideanEstimator::bundleAdjustement ( )

Run a bundle adjustment using every computed cameras and every computed 3D points

Definition at line 148 of file EuclideanEstimator.cpp.

Referenced by computeReconstruction().

4.5.3.3 bool OpencvSfM::EuclideanEstimator::cameraResection (unsigned int *image*)

Find the position of a new camera

Parameters

image	index of the wanted camera

Definition at line 452 of file EuclideanEstimator.cpp.

Referenced by computeReconstruction().

4.5.3.4 void OpencvSfM::EuclideanEstimator::computeReconstruction ( )

comptue cameras and structure if intra parameters are known.

Definition at line 744 of file EuclideanEstimator.cpp.

4.5.3.5 void OpencvSfM::EuclideanEstimator::initialReconstruction (int *image1,* int *image2*)

Create a new Euclidean reconstruction using matches between two images

Parameters

image1	index of the first image
image2	index of the second image

Definition at line 652 of file EuclideanEstimator.cpp.

Referenced by computeReconstruction().

4.5.3.6 void OpencvSfM::EuclideanEstimator::viewEstimation ( )

Show this estimation

Definition at line 868 of file EuclideanEstimator.cpp.

Referenced by computeReconstruction().

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

- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/-EuclideanEstimator.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/-EuclideanEstimator.cpp

4.6 OpencvSfM::ImageLink Struct Reference

This structure store an image link (two image ids)...

`#include <TracksOfPoints.h>`

Public Attributes

- int imgSrc

  index of first image
- int imgDest

  index of second image

4.6.1 Detailed Description

This structure store an image link (two image ids)...

Definition at line 241 of file TracksOfPoints.h.

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

- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/TracksOfPoints.h

4.7 OpencvSfM::ImagesGraphConnection Class Reference

This class modelizes the images graph connections.

```
#include <TracksOfPoints.h>
```

Public Member Functions

- ImagesGraphConnection ()
- bool isGraphCreated (int nbImages)
- void initStructure (int nb_images)
- void addLink (int first_image, int second_image)
- int getNumbersOfLinks (int first_image, int second_image)
- int getHighestLink (int &first_image, int &second_image, int max_number=1e9)
- void getOrderedLinks (std::vector< ImageLink > &outList, int min_number=0, int max_number=1e9)
- void getImagesRelatedTo (int first_image, std::vector< ImageLink > &outList, int min_number=0, int max_number=1e9)

Protected Member Functions

- void orderedIdx (int i1, int i2, int idx[2])

Protected Attributes

- cv::SparseMat images_graph_

4.7.1 Detailed Description

This class modelizes the images graph connections.

Definition at line 250 of file TracksOfPoints.h.

4.7.2 Constructor & Destructor Documentation

4.7.2.1 OpencvSfM::ImagesGraphConnection::ImagesGraphConnection ( ) [inline]

Create an empty image graph

Definition at line 283 of file TracksOfPoints.h.

4.7.3 Member Function Documentation

4.7.3.1 void OpencvSfM::ImagesGraphConnection::addLink (int *first_image,* int *second_image*) [inline]

Add a new link between two images

Parameters

first_image	first image
second_image	second image

Definition at line 312 of file TracksOfPoints.h.

Referenced by OpencvSfM::SequenceAnalyzer::constructImagesGraph().

4.7.3.2 int OpencvSfM::ImagesGraphConnection::getHighestLink (int & *first_image,* int & *second_image,* int *max_number* = 1e9)

get the highest link

Parameters

first_image	[out] first image
second_image	[out] second image
max_number	[in] maximum allowed links between images

Returns

numbers of links between first image and second image

Definition at line 441 of file TracksOfPoints.cpp.

Referenced by OpencvSfM::EuclideanEstimator::computeReconstruction().

4.7.3.3 void OpencvSfM::ImagesGraphConnection::getImagesRelatedTo (int *first_image,* std::vector< **ImageLink** > & *outList,* int *min_number* = 0, int *max_number* = 1e9)

get the related images to the first parameter

Parameters

first_image	[in] first image index
outList	[in/out] ordered vector of links between images
min_number	minimum allowed links between images
max_number	maximum allowed links between images

Definition at line 492 of file TracksOfPoints.cpp.

Referenced by OpencvSfM::EuclideanEstimator::computeReconstruction().

4.7.3.4 int OpencvSfM::ImagesGraphConnection::getNumbersOfLinks (int *first_image,* int *second_image*) [inline]

get the numbers of links between two images

Parameters

first_image	first image
second_image	second image

Returns

numbers of links between first image and second image

Definition at line 324 of file TracksOfPoints.h.

4.7.3.5 void OpencvSfM::ImagesGraphConnection::getOrderedLinks (std::vector< **ImageLink** > & *outList,* int *min_number* = 0, int *max_number* = 1e9)

get the highest link

Parameters

outList	[out] ordered vector of links between images
min_number	minimum allowed links between images
max_number	maximum allowed links between images

Definition at line 465 of file TracksOfPoints.cpp.

Referenced by OpencvSfM::EuclideanEstimator::computeReconstruction().

4.7.3.6 void OpencvSfM::ImagesGraphConnection::initStructure (int *nb_images*) [inline]

Prepare this structure to store the graph of correspondances

Parameters

nb_images	number of images to store

Definition at line 301 of file TracksOfPoints.h.

Referenced by OpencvSfM::SequenceAnalyzer::constructImagesGraph().

4.7.3.7 bool OpencvSfM::ImagesGraphConnection::isGraphCreated (int *nbImages*) [inline]

Use this function to test if the graph is already builded

Parameters

nbImages	number of images the graph should store

Returns

true if graph is build

Definition at line 290 of file TracksOfPoints.h.

4.7.3.8 void OpencvSfM::ImagesGraphConnection::orderedIdx (int *i1,* int *i2,* int *idx[2]*) [inline, protected]

Use this function to create an ordered image index:

Parameters

i1	[in] first image index
i2	[in] second image index
idx	[out] index of this image link where idx[0]<idx[1]

Definition at line 266 of file TracksOfPoints.h.

4.7.4 Member Data Documentation

4.7.4.1 cv::SparseMat OpencvSfM::ImagesGraphConnection::images_graph_ [protected]

Sparse upper triangular matrix for image graph. (i,j) value represent the numbers of points matches between image i and j. of course (i,j) equal (j,i) so only (i,j) with i<j

are stored.

Definition at line 258 of file TracksOfPoints.h.

Referenced by getHighestLink(), getImagesRelatedTo(), and getOrderedLinks().

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

- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/TracksOfPoints.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/TracksOfPoints.cpp

4.8 OpencvSfM::MatchingThread Struct Reference

This struct is used by boost::thread object to compute match. I used some semaphore to ensure the matching process work well.

`#include <Boost_Matching.h>`

Public Member Functions

- CREATE_STATIC_MUTEX (thread_concurr)

  Used to start as many thread as processors.
- CREATE_STATIC_MUTEX (thread_unicity)

  Used around critical sections.
- MatchingThread (cv::Ptr< SequenceAnalyzer > seq_analyser, unsigned int i, std::vector< cv::Ptr< PointsToTrack > >::iterator matches_it)
- void operator() ()

Public Attributes

- unsigned int i

  Index of source image. This image will be matched against every other.
- std::vector< cv::Ptr < PointsToTrack > >::iterator matches_it

  iterator of every Points for track (points of other images to match)
- cv::Ptr< SequenceAnalyzer > seq_analyser

  This object contains every sequence related info (images, points, tracks...)

Static Public Attributes

- static std::vector< cv::Ptr < PointsToTrack > >::iterator end_matches_it

  End of list images of points. It's the same for every thread, so set once for every thread before runing computation.
- static std::vector< cv::Mat > masks

List of mask to hide some points in the matching computation.

- static unsigned int mininum_points_matches = 50

  Minimum matches between two images to accept the matches.

- static PointsMatcher ∗ match_algorithm = NULL

  The algorithm to use for matching.

4.8.1 Detailed Description

This struct is used by boost::thread object to compute match. I used some semaphore to ensure the matching process work well.

Definition at line 17 of file Boost_Matching.h.

4.8.2 Constructor & Destructor Documentation

4.8.2.1 OpencvSfM::MatchingThread::MatchingThread (cv::Ptr< SequenceAnalyzer > *seq_analyser*, unsigned int *i*, std::vector< cv::Ptr< PointsToTrack > >::iterator *matches_it*)

Constructor of a thread.

Parameters

seq_analyser	the sequence related infos
i	Index of source image. This image will be matched against every other
matches_it	iterator of every Points for track (points of other images to match)

Definition at line 20 of file Boost_Matching.cpp.

4.8.3 Member Function Documentation

4.8.3.1 void OpencvSfM::MatchingThread::operator() ( )

Thread implementation...

Definition at line 29 of file Boost_Matching.cpp.

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

- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/Boost_Matching.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/Boost_Matching.cpp

4.9 OpencvSfM::MotionProcessor Class Reference

This class try to create a commun interface for files loading. Indeed, if you want to use webcam, avi file of list of files, you will have to do some annoying processing, like iterate the different files of the directory. With MotionProcessor, you can now use a folder of image the same way you use a webcam or a video file.

```
#include <MotionProcessor.h>
```

Public Member Functions

- bool isBidirectional ()
- bool setInputSource (int idWebCam)
- bool setInputSource (std::vector< std::string > list_images)
- bool setInputSource (std::string nameOfFile, TypeOfMotionProcessor inputType=IS_SINGLE_FILE)
- bool setInputSource (std::string prefix, std::string suffix, int startNumber=0)
- cv::Mat getFrame ()
- bool setProperty (int idProp, double value)
- double getProperty (int idProp)

Protected Attributes

- TypeOfMotionProcessor type_of_input_

  This attribut is used to know which type is the input (webcam, video file, list of file or just one image)
- cv::VideoCapture capture_

  When the camera is attached to an avi file or webcam, this will be usefull to get frame...
- std::vector< std::string > nameOfFiles_

  If the motion processor use directory as input, we store here the names of files.
- std::string sourceName_
- std::string suffix_
- unsigned int pos_in_loading_process_
- unsigned int numFrame_

  When the camera is attached to a list of file, numFrame_ will be used to know how many frames we have take.
- int wantedWidth_

  if below 0, represent the wanted width of Mat returned by getFrame( );
- int wantedHeight_

  if below 0, represent the wanted height of Mat returned by getFrame( );
- uchar convertToRGB_

4.9.1 Detailed Description

This class try to create a commun interface for files loading. Indeed, if you want to use webcam, avi file of list of files, you will have to do some annoying processing, like iterate the different files of the directory. With MotionProcessor, you can now use a folder of image the same way you use a webcam or a video file.

The class is still in development as the way to open folder is not really clear... The easy way would be to use "dirent.h" header, but the easy thing is not always the best thing...

Definition at line 28 of file MotionProcessor.h.

4.9.2 Member Function Documentation

4.9.2.1 cv::Mat OpencvSfM::MotionProcessor::getFrame ( )

use this method if you want to get a picture from this motion handler

Returns

> The current frame. If the video is finished, the Mat returned is not usable! Test if the matrix is empty before using it!

Definition at line 131 of file MotionProcessor.cpp.

Referenced by OpencvSfM::SequenceAnalyzer::SequenceAnalyzer().

4.9.2.2 double OpencvSfM::MotionProcessor::getProperty (int *idProp*)

use this method to get actual properties of pictures retrived by this MotionProcessor. the properties are the same than VideoCapture (see `http://opencv.willowgarage.com/documentation/cpp/reading_and_writing_images_and_video.html#cv-videocapture-get`)

Parameters

idProp	Property identifier

Returns

> the value of the property

Definition at line 302 of file MotionProcessor.cpp.

4.9.2.3 bool OpencvSfM::MotionProcessor::isBidirectional ( ) `[inline]`

Use this function to know if this flux is bidirectional (i.e. frames can be iterate randomly) Can be used to know if the sequence is finite

Returns

true is you can access to frames randomly, false else

Definition at line 69 of file MotionProcessor.h.

Referenced by OpencvSfM::SequenceAnalyzer::SequenceAnalyzer().

4.9.2.4 bool OpencvSfM::MotionProcessor::setInputSource (int *idWebCam*)

You can attach this motion handler to a webcam use this method to set it as the input source!

Parameters

idWebCam	id of the webcam

Returns

true if input source opened without problems

Definition at line 57 of file MotionProcessor.cpp.

4.9.2.5 bool OpencvSfM::MotionProcessor::setInputSource (std::vector< std::string > *list_images*)

You can attach this motion handler to a list of picture use this method to set it as the input source!

Parameters

list_images	list of pictures' names

Returns

true if input source opened without problems

Definition at line 73 of file MotionProcessor.cpp.

4.9.2.6 bool OpencvSfM::MotionProcessor::setInputSource (std::string *nameOfFile*, TypeOfMotionProcessor *inputType* = IS_SINGLE_FILE)

You can attach this motion handler to a video file or a single picture. use this method to set it as the input source!

Parameters

nameOfFile	name of the media file (picture or avi movie)
inputType	type of input (can be either IS_DIRECTORY, IS_VIDEO or IS_SINGLE_FILE)

Returns

true if input source opened without problems

Definition at line 81 of file MotionProcessor.cpp.

4.9.2.7 bool OpencvSfM::MotionProcessor::setInputSource (std::string *prefix*, std::string *suffix*, int *startNumber* = 0)

You can attach this motion handler to a list of file. use this method to set the input source! For example, if the files are img1.jpg, img2.jpg, ... img125.jpg, prefix will be equal to "img", suffix to ".jpg" and startNumber equal to 1

Parameters

prefix	the part of the files names which stay the same (img)
suffix	the type of the files (.jpg for instance)
startNumber	the first number to use...

Returns

true if input source opened without problems

Definition at line 120 of file MotionProcessor.cpp.

4.9.2.8 bool OpencvSfM::MotionProcessor::setProperty (int *idProp*, double *value*)

use this method to change the properties of pictures retrived by this MotionProcessor. the properties are the same than VideoCapture (see `http://opencv.willowgarage.com/documentation/cpp/reading_and_writing_images_and_video.html#cv-videocapture-get`)

Parameters

idProp	Property identifier
value	new value of the property

Definition at line 227 of file MotionProcessor.cpp.

Referenced by OpencvSfM::SequenceAnalyzer::SequenceAnalyzer().

4.9.3 Member Data Documentation

4.9.3.1 uchar OpencvSfM::MotionProcessor::convertToRGB_ [protected]

if >0 the loaded image is forced to be a 3-channel color image if =0 the loaded image is forced to be grayscale if <0 the loaded image will be loaded as-is

Definition at line 58 of file MotionProcessor.h.

4.9.3.2 unsigned int OpencvSfM::MotionProcessor::pos_in_loading_process_ [protected]

When the camera is attached to a list of file, pos_in_loading_process_ will be used to store actual number of image (not always the same than numFrame_).

Definition at line 48 of file MotionProcessor.h.

4.9.3.3 std::string OpencvSfM::MotionProcessor::sourceName_ [protected]

When the camera is attached to a list of file, sourceName_ will be used to store name of the prefix. For example, if the files are img1.jpg, img2.jpg, ... img125.jpg, sourceName_ will be equal to img

Definition at line 38 of file MotionProcessor.h.

4.9.3.4 std::string OpencvSfM::MotionProcessor::suffix_ [protected]

When the camera is attached to a list of file, suffix_ will be used to store name of the suffix. For example, if the files are img1.jpg, img2.jpg, ... img125.jpg, suffix_ will be equal to .jpg

Definition at line 43 of file MotionProcessor.h.

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

- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/MotionProcessor.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/MotionProcessor.cpp

4.10 OpencvSfM::mapping::Point Struct Reference

This structure will handle conversions between OpenCV and PCL data.

`#include <PCL_mapping.h>`

Public Member Functions

- Point (const Point &otherP)
- Point & operator= (const Point &otherP)
- Point ()
- Point (float *data, int sizeOfBuf=4)
- template<typename Type , int size>
Point (cv::Vec< Type, size > &v)
- template<typename Type , int size>
Point (cv::Matx< Type, size, 1 > &matX)

- Point (cv::KeyPoint &kp)
- Point (pcl::PointXY &pXY)
- Point (pcl::PointXYZ &pXYZ)
- Point (pcl::PointXYZI &pXYZi)
- Point (pcl::InterestPoint &iP)
- Point (pcl::PointWithRange &pPWR)
- Point (pcl::PointXYZRGBA &pXYZ1)
- Point (pcl::PointXYZRGB &pXYZ2)
- ~Point ()
- template<typename Type , int size>
  operator cv::Matx< Type, size, 1 > & ()
- template<typename Type , int size>
  operator cv::Vec< Type, size > & ()
- template<typename Type >
  operator cv::Point3_< Type > & ()
- operator cv::KeyPoint & ()
- operator pcl::PointXY & ()
- operator pcl::PointXYZ & ()
- operator pcl::PointXYZI & ()
- operator pcl::InterestPoint & ()
- operator pcl::PointWithRange & ()
- operator pcl::PointXYZRGBA & ()
- operator pcl::PointXYZRGB & ()

Public Attributes

- float * data_

  values of datas
- unsigned char size_of_data

  Size of data buffer.
- bool should_remove

  Used to know if data were allocated.

4.10.1 Detailed Description

This structure will handle conversions between OpenCV and PCL data.

Definition at line 140 of file PCL_mapping.h.

4.10.2 Constructor & Destructor Documentation

4.10.2.1 OpencvSfM::mapping::Point::Point (const Point & *otherP*) [inline]

Copy constructor (deep copy!)

Definition at line 149 of file PCL_mapping.h.

4.10.2.2 OpencvSfM::mapping::Point::Point ( ) [inline]

Init data using the max size of points in both library (8 floats)

Definition at line 181 of file PCL_mapping.h.

4.10.2.3 OpencvSfM::mapping::Point::Point (float ∗ *data*, int *sizeOfBuf* = 4) [inline]

Init data using previously allocated buffer

Parameters

data	values of point to convert
sizeOfBuf	in number of float, the size of point

Definition at line 188 of file PCL_mapping.h.

4.10.2.4 template<typename Type , int size> OpencvSfM::mapping::Point::Point (cv::Vec< Type, size > & *v*) [inline]

Init data using an opencv vector

Parameters

v	input vector

Definition at line 197 of file PCL_mapping.h.

4.10.2.5 template<typename Type , int size> OpencvSfM::mapping::Point::Point (cv::Matx< Type, size, 1 > & *matX*) [inline]

Init data using an opencv matrix

Parameters

matX	input matrix

Definition at line 212 of file PCL_mapping.h.

4.10.2.6 OpencvSfM::mapping::Point::Point (cv::KeyPoint & *kp*) [inline]

Init data using an opencv KeyPoint

Parameters

kp	input KeyPoint

Definition at line 226 of file PCL_mapping.h.

4.10.2.7 OpencvSfM::mapping::Point::Point (pcl::PointXY & *pXY*) [inline]

Init data using a PCL KeyPoint

Parameters

pXY	input KeyPoint

Definition at line 234 of file PCL_mapping.h.

4.10.2.8 OpencvSfM::mapping::Point::Point (pcl::PointXYZ & *pXYZ*) [inline]

Init data using a PCL KeyPoint

Parameters

pXYZ	input KeyPoint

Definition at line 241 of file PCL_mapping.h.

4.10.2.9 OpencvSfM::mapping::Point::Point (pcl::PointXYZI & *pXYZi*) [inline]

Init data using a PCL KeyPoint

Parameters

pXYZi	input KeyPoint

Definition at line 246 of file PCL_mapping.h.

4.10.2.10 OpencvSfM::mapping::Point::Point (pcl::InterestPoint & *iP*) [inline]

Init data using a PCL KeyPoint

Parameters

iP	input KeyPoint

Definition at line 251 of file PCL_mapping.h.

4.10.2.11 OpencvSfM::mapping::Point::Point (pcl::PointWithRange & *pPWR*) [inline]

Init data using a PCL KeyPoint

Parameters

pPWR	input KeyPoint

Definition at line 256 of file PCL_mapping.h.

4.10.2.12 OpencvSfM::mapping::Point::Point (pcl::PointXYZRGBA & *pXYZ1*) [inline]

Init data using a PCL KeyPoint

Parameters

pXYZ1	input KeyPoint

Definition at line 261 of file PCL_mapping.h.

4.10.2.13 OpencvSfM::mapping::Point::Point (pcl::PointXYZRGB & *pXYZ2*) [inline]

Init data using a PCL KeyPoint

Parameters

pXYZ2	input KeyPoint

Definition at line 266 of file PCL_mapping.h.

4.10.2.14 OpencvSfM::mapping::Point::∼Point ( ) [inline]

Destructor of PCL point convertor. Free allocated data if needed.

Definition at line 272 of file PCL_mapping.h.

4.10.3 Member Function Documentation

4.10.3.1 OpencvSfM::mapping::Point::operator cv::KeyPoint & ( ) [inline]

Conversions operators to opencv KeyPoint:

Definition at line 302 of file PCL_mapping.h.

4.10.3.2 template<typename Type , int size> OpencvSfM::mapping::Point::operator cv::Matx< Type, size, 1 > & ( ) [inline]

Conversions operators to opencv Matx:

Definition at line 279 of file PCL_mapping.h.

4.10.3.3 template<typename Type > OpencvSfM::mapping::Point::operator cv::Point3_< Type > & ( ) [inline]

Conversions operators to opencv Point3_:

Definition at line 295 of file PCL_mapping.h.

4.10.3.4 template<typename Type , int size> OpencvSfM::mapping::Point::operator cv::Vec< Type, size > & ( ) [inline]

Conversions operators to opencv Vec:

Definition at line 287 of file PCL_mapping.h.

4.10.3.5 OpencvSfM::mapping::Point::operator pcl::InterestPoint & ( ) [inline]

Conversions operators to PCL KeyPoint:

Definition at line 327 of file PCL_mapping.h.

4.10.3.6 OpencvSfM::mapping::Point::operator pcl::PointWithRange & ( ) [inline]

Conversions operators to PCL KeyPoint:

Definition at line 333 of file PCL_mapping.h.

4.10.3.7 OpencvSfM::mapping::Point::operator pcl::PointXY & ( ) [inline]

Conversions operators to PCL KeyPoint:

Definition at line 309 of file PCL_mapping.h.

4.10.3.8 OpencvSfM::mapping::Point::operator pcl::PointXYZ & ( ) [inline]

Conversions operators to PCL KeyPoint:

Definition at line 315 of file PCL_mapping.h.

4.10.3.9 OpencvSfM::mapping::Point::operator pcl::PointXYZI & ( ) [inline]

Conversions operators to PCL KeyPoint:

Definition at line 321 of file PCL_mapping.h.

4.10.3.10 OpencvSfM::mapping::Point::operator pcl::PointXYZRGB & ( ) [inline]

Conversions operators to PCL KeyPoint:

Definition at line 345 of file PCL_mapping.h.

4.10.3.11 OpencvSfM::mapping::Point::operator pcl::PointXYZRGBA & ( ) [inline]

Conversions operators to PCL KeyPoint:

Definition at line 339 of file PCL_mapping.h.

4.10.3.12 Point& OpencvSfM::mapping::Point::operator= (const Point & *otherP*) [inline]

operator =(deep copy!)

Definition at line 157 of file PCL_mapping.h.

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

- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/PCL_mapping.h

4.11 OpencvSfM::PointOfView Class Reference

This class represent the 3D position of the device which take the pictures. The role of the class is to store everything related to the filed of view: picture, 3D position, points, matches and 3D points.

```
#include <PointOfView.h>
```

Public Member Functions

- PointOfView (cv::Ptr< Camera > device, cv::Mat rotation=cv::Mat::eye(3, 3, CV_64F), cv::Vec3d translation=cv::Vec3d(0.0, 0.0, 0.0))
- PointOfView (cv::Mat projection_matrix)
- virtual ∼PointOfView (void)
- cv::Ptr< Camera > getIntraParameters () const
- virtual std::vector< cv::Vec2d > project3DPointsIntoImage (std::vector< TrackOfPoints > points) const
- virtual std::vector< cv::Vec2d > project3DPointsIntoImage (std::vector< cv::Vec3d > points) const
- virtual cv::Vec2d project3DPointIntoImage (cv::Vec3d point) const
- virtual bool pointInFrontOfCamera (cv::Vec4d point) const
- virtual cv::Mat getProjectionMatrix () const
- cv::Mat getRotationMatrix () const
- virtual void setRotationMatrix (cv::Mat newRot)
- cv::Mat getTranslationVector () const
- virtual void setTranslationVector (cv::Mat newVect)
- void rotationAroundX (double angle)
- void rotationAroundY (double angle)
- void rotationAroundZ (double angle)

Static Public Member Functions

- static cv::Ptr< PointOfView > read (const cv::FileNode &node)
- static void write (cv::FileStorage &fs, const PointOfView &points)

Protected Attributes

- cv::Mat rotation_

  Rotation matrix R (data is stored into projection_matrix_)
- cv::Mat translation_

  Translation vector t (Matrix instead of vector because data is stored into projection_-matrix_)
- cv::Mat projection_matrix_

  redundancy but speed improvement
- cv::Ptr< Camera > device_

  intra parameters and distortion coefs
- unsigned char config_

  This attribut is used to know what we should estimate... If equal to 0, nothing should be estimated...

4.11.1 Detailed Description

This class represent the 3D position of the device which take the pictures. The role of the class is to store everything related to the filed of view: picture, 3D position, points, matches and 3D points.

We use the so-called pinhole camera model. That is, a scene view is formed by projecting 3D points into the image plane using a perspective transformation. Usual notation says that a point [u,v] from an image is related to the point [X,Y,Z] using the following notation :

$$s\begin{bmatrix}u\\v\\1\end{bmatrix}=\begin{bmatrix}f_x&0&c_x\\0&f_y&c_y\\0&0&1\end{bmatrix}\begin{bmatrix}r_{11}&r_{12}&r_{13}&t_1\\r_{21}&r_{22}&r_{23}&t_2\\r_{31}&r_{32}&r_{33}&t_3\end{bmatrix}\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$$

This leads to the following relation between local coordinates and global ones:

$$\begin{bmatrix}x\\y\\z\end{bmatrix}=R\begin{bmatrix}X\\Y\\Z\end{bmatrix}+t$$

$$x'=x/z$$
$$y'=y/z$$

Definition at line 44 of file PointOfView.h.

4.11.2 Constructor & Destructor Documentation

4.11.2.1 OpencvSfM::PointOfView::PointOfView (cv::Ptr< Camera > *device*, cv::Mat *rotation* = `cv::Mat::eye( 3, 3, CV_64F )`, cv::Vec3d *translation* = `cv::Vec3d( 0.0,0.0,0.0 )`)

To create a point of view, we need two things : a camera, and a point (with orientation). Here we give an address of a Camera, and the file name of the picture. If we have more informations, we can use the last parameters...

Parameters

device	address of existing Camera. This camera can be calibrated or not...
rotation	Matrix of the known rotation (optional)...
translation	Vector of the known translation (optional)...

Definition at line 34 of file PointOfView.cpp.

Referenced by read().

4.11.2.2 OpencvSfM::PointOfView::PointOfView (cv::Mat *projection_matrix*)

To create a point of view using a projection matrix. (will create a Pinhole Camera without distortion parameters) We will extract intra, rotation and translation from this projection matrix.

Parameters

projection_-matrix	Projection matrix of the camera

Definition at line 57 of file PointOfView.cpp.

4.11.2.3 OpencvSfM::PointOfView::∼PointOfView (void) `[virtual]`

Destructor of PointOfView, release all vectors... TODO: define how we should release the vectors...

Definition at line 98 of file PointOfView.cpp.

4.11.3 Member Function Documentation

4.11.3.1 cv::Ptr<Camera> OpencvSfM::PointOfView::getIntraParameters ( ) const `[inline]`

use this function to get acces to the camera parameters

Returns

camera matrix

Definition at line 80 of file PointOfView.h.

Referenced by OpencvSfM::Visualizer::addCamera(), and OpencvSfM::EuclideanEstimator::addNewPointOfView().

4.11.3.2 cv::Mat OpencvSfM::PointOfView::getProjectionMatrix ( ) const [virtual]

This method return the intra parameters of the camera

Returns

Matrix K of intra parameters

Definition at line 227 of file PointOfView.cpp.

4.11.3.3 cv::Mat OpencvSfM::PointOfView::getRotationMatrix ( ) const [inline]

Use this method to get the rotation matrix of this camera

Returns

rotation matrix of this camera

Definition at line 116 of file PointOfView.h.

Referenced by OpencvSfM::Visualizer::addCamera(), and OpencvSfM::EuclideanEstimator::addNewPointOfView().

4.11.3.4 cv::Mat OpencvSfM::PointOfView::getTranslationVector ( ) const [inline]

Use this method to get the translation vector of this camera

Returns

translation vector of this camera

Definition at line 135 of file PointOfView.h.

Referenced by OpencvSfM::Visualizer::addCamera(), and OpencvSfM::EuclideanEstimator::addNewPointOfView().

4.11.3.5 bool OpencvSfM::PointOfView::pointInFrontOfCamera (cv::Vec4d *point*) const [virtual]

This method test is 3D point is in front of Camera (can be view with the camera)

Parameters

point	3D point in world coordinates (homogeneous, that is 4 values).

Returns

true if point can be seen with this point of view

Definition at line 216 of file PointOfView.cpp.

4.11.3.6 cv::Vec2d OpencvSfM::PointOfView::project3DPointIntoImage (cv::Vec3d *point*) const `[virtual]`

This method can convert 3D point from world coordinates to 2D point in pixel image coordinates

Parameters

point	3D point in world coordinates.

Returns

2D point in pixel image coordinates.

Definition at line 117 of file PointOfView.cpp.

4.11.3.7 std::vector< cv::Vec2d > OpencvSfM::PointOfView::project3DPointsIntoImage (std::vector< TrackOfPoints > *points*) const `[virtual]`

This method can convert 3D points from world coordinates to 2D points in pixel image coordinates

Parameters

points	3D points in world coordinates.

Returns

2D points in pixel image coordinates.

Definition at line 178 of file PointOfView.cpp.

4.11.3.8 std::vector< cv::Vec2d > OpencvSfM::PointOfView::project3DPointsIntoImage (std::vector< cv::Vec3d > *points*) const `[virtual]`

This method can convert 3D points from world coordinates to 2D points in pixel image coordinates

Parameters

points	3D points in world coordinates.

Returns

2D points in pixel image coordinates.

Definition at line 145 of file PointOfView.cpp.

4.11.3.9 cv::Ptr< PointOfView > OpencvSfM::PointOfView::read (const cv::FileNode & *node*) [static]

Create a new camera's point of view from a YAML file.

Parameters

node	Previously opened YAML file node

Definition at line 233 of file PointOfView.cpp.

4.11.3.10 void OpencvSfM::PointOfView::rotationAroundX (double *angle*) [inline]

Rotate this camera around X axis

Parameters

angle	of rotation

Definition at line 153 of file PointOfView.h.

4.11.3.11 void OpencvSfM::PointOfView::rotationAroundY (double *angle*) [inline]

Rotate this camera around Y axis

Parameters

angle	of rotation

Definition at line 163 of file PointOfView.h.

4.11.3.12 void OpencvSfM::PointOfView::rotationAroundZ (double *angle*) [inline]

Rotate this camera around Z axis

Parameters

angle	of rotation

Definition at line 173 of file PointOfView.h.

4.11.3.13 virtual void OpencvSfM::PointOfView::setRotationMatrix (cv::Mat *newRot*) [inline, virtual]

Use this method to change the rotation matrix of this camera

Parameters

newRot	new rotation matrix

Definition at line 124 of file PointOfView.h.

4.11.3.14 virtual void OpencvSfM::PointOfView::setTranslationVector (cv::Mat *newVect*) [inline, virtual]

Use this method to change the translation vector of this camera

Parameters

newVect	new translation vector

Definition at line 143 of file PointOfView.h.

4.11.3.15 void OpencvSfM::PointOfView::write (cv::FileStorage & *fs*, const PointOfView & *points*) [static]

Save the camera's point of view into a YAML file.

Parameters

fs	Previously opened YAML file node
points	sequence to save...

Definition at line 261 of file PointOfView.cpp.

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

- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/PointOfView.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/PointOfView.cpp

4.12 OpencvSfM::PointsMatcher Class Reference

A class used for matching descriptors that can be described as vectors in a finite-dimensional space.

```
#include <PointsMatcher.h>
```

Public Member Functions

- PointsMatcher (const cv::Ptr< cv::DescriptorMatcher > &matcher)
- PointsMatcher (const PointsMatcher &copy)
- virtual ∼PointsMatcher ()
- virtual void add (cv::Ptr< PointsToTrack > pointCollection)
- virtual void clear ()
- virtual void train ()
- virtual bool isMaskSupported ()
- virtual bool empty () const
- virtual cv::Ptr< PointsMatcher > clone (bool emptyTrainData=true)
- virtual void match (cv::Ptr< PointsToTrack > queryPoints, std::vector< cv::DMatch > &matches, const std::vector< cv::Mat > &masks=std::vector< cv::Mat >())
- virtual void knnMatch (cv::Ptr< PointsToTrack > queryPoints, std::vector< std::vector< cv::DMatch > > &matches, int k, const std::vector< cv::Mat > &masks=std::vector< cv::Mat >(), bool compactResult=true)
- virtual void radiusMatch (cv::Ptr< PointsToTrack > queryPoints, std::vector< std::vector< cv::DMatch > > &matches, float maxDistance, const std::vector< cv::Mat > &masks=std::vector< cv::Mat >(), bool compactResult=true)
- virtual void crossMatch (cv::Ptr< PointsMatcher > otherMatcher, std::vector< cv::DMatch > &matches, const std::vector< cv::Mat > &masks=std::vector< cv::Mat >())
- const cv::KeyPoint & getKeypoint (int numKey) const

Static Public Member Functions

- static cv::Ptr< PointsMatcher > create (std::string match_algo)
- static void drawMatches (const cv::Mat &img1, const std::vector< cv::KeyPoint > &keypoints1, const std::vector< cv::KeyPoint > &keypoints2, const std::vector< cv::DMatch > &matches1to2, cv::Mat &outImg, const cv::Scalar &matchColor=cv::Scalar::all(-1), const cv::Scalar &singlePointColor=cv::Scalar::all(-1), const std::vector< char > &matchesMask=std::vector< char >(), int flags=cv::DrawMatchesFlags::DEFAULT)
- static void read (const cv::FileNode &node, PointsMatcher &points)
- static void write (cv::FileStorage &fs, const PointsMatcher &points)

Protected Attributes

- cv::Ptr< cv::DescriptorMatcher > matcher_

  Algorithm used to find matches...
- std::vector< cv::Ptr < PointsToTrack > > pointCollection_

  Vector of points used to compute matches...

4.12.1 Detailed Description

A class used for matching descriptors that can be described as vectors in a finite-dimensional space.

Any Matcher that inherit from DescriptorMatcher can be used (For example, you can use FlannBasedMatcher or BruteForceMatcher).

Definition at line 17 of file PointsMatcher.h.

4.12.2 Constructor & Destructor Documentation

4.12.2.1 OpencvSfM::PointsMatcher::PointsMatcher (const cv::Ptr< cv::DescriptorMatcher > & *matcher*)

Constructor. Need a matcher algorithm...

Parameters

matcher	Ptr on a matcher. See for available matcher: `http://opencv.willowgarage.com/documentation/cpp/features2d_common_interfaces_of_descriptor_matchers.html#descriptormatcher`

Definition at line 18 of file PointsMatcher.cpp.

Referenced by clone().

4.12.2.2 OpencvSfM::PointsMatcher::PointsMatcher (const **PointsMatcher** & *copy*)

Copy constructor.

Definition at line 25 of file PointsMatcher.cpp.

4.12.2.3 OpencvSfM::PointsMatcher::∼PointsMatcher (void) `[virtual]`

Destructor...

Definition at line 32 of file PointsMatcher.cpp.

4.12.3 Member Function Documentation

4.12.3.1 void OpencvSfM::PointsMatcher::add (cv::Ptr< PointsToTrack > *pointCollection*) [virtual]

Use this function to add data used to find matches

Parameters

pointCollection	points computed using various methods. Please be carful to get compatible points (that is with descriptors if matcher need some)

Definition at line 43 of file PointsMatcher.cpp.

4.12.3.2 void OpencvSfM::PointsMatcher::clear ( ) [virtual]

If needed, you can clear the training data using this method.

Definition at line 50 of file PointsMatcher.cpp.

4.12.3.3 Ptr< PointsMatcher > OpencvSfM::PointsMatcher::clone (bool *emptyTrainData* = true) [virtual]

Clone the matcher.

Parameters

emptyTrainData	IIf emptyTrainData is false the method create deep copy of the object, i.e. copies both parameters and train data. If emptyTrainData is true the method create object copy with current parameters but with empty train data..

Returns

An other PointsMatcher instance

Definition at line 159 of file PointsMatcher.cpp.

Referenced by OpencvSfM::MatchingThread::operator()().

4.12.3.4 static cv::Ptr<PointsMatcher> OpencvSfM::PointsMatcher::create (std::string *match_algo*) [inline, static]

Use this function to create a point matcher using the name of a matching algorithm (see http://opencv.willowgarage.com/documentation/cpp/features2d-_common_interfaces_of_descriptor_matchers.html)

Parameters

match_algo	name of the wanted algorithm

Returns

Definition at line 46 of file PointsMatcher.h.

4.12.3.5 void OpencvSfM::PointsMatcher::crossMatch (cv::Ptr< PointsMatcher > *otherMatcher*, std::vector< cv::DMatch > & *matches*, const std::vector< cv::Mat > & *masks* = `std::vector<cv::Mat>( )`) `[virtual]`

Using an other matchers given in parameters, recompute a matching in inverse order and keep only matches which are two-ways.

Parameters

otherMatcher	Query set of points and descriptors.
matches	First guess of matches... Will be updated to contain only two-way matches (can be empty).
masks	specifying permissible matches between input query and train matrices of descriptors.

Definition at line 170 of file PointsMatcher.cpp.

4.12.3.6 void OpencvSfM::PointsMatcher::drawMatches (const cv::Mat & *img1*, const std::vector< cv::KeyPoint > & *keypoints1*, const std::vector< cv::KeyPoint > & *keypoints2*, const std::vector< cv::DMatch > & *matches1to2*, cv::Mat & *outImg*, const cv::Scalar & *matchColor* = `cv::Scalar::all( -1 )`, const cv::Scalar & *singlePointColor* = `cv::Scalar::all( -1 )`, const std::vector< char > & *matchesMask* = `std::vector<char>( )`, int *flags* = `cv::DrawMatchesFlags::DEFAULT`) `[static]`

This function draw keypoints and matches. Contrary to cv::drawMatches, only the first image is used to draw matches...

Parameters

img1	First source image.
keypoints1	Keypoints from first source image.
keypoints2	Keypoints from second source image.
matches1to2	Matches from first image to second one, i.e. keypoints1[i] has corresponding point keypoints2[matches[i]] .
outImg	Output image. Its content depends on flags value what is drawn in output image. See below possible flags bit values.

matchColor	Color of matches (lines and connected keypoints). If matchColor==Scalar::all(-1) color will be generated randomly.
singlePointColor	Color of single keypoints (circles), i.e. keypoints not having the matches. If singlePointColor==Scalar::all(-1) color will be generated randomly.
matchesMask	Mask determining which matches will be drawn. If mask is empty all matches will be drawn.
flags	Each bit of flags sets some feature of drawing. Possible flags bit values is defined by DrawMatchesFlags , see http://opencv.willowgarage.com/documentation/cpp/features2d_drawing_function_of_keypoints_and_matches.html#cv-drawmatches.

Definition at line 209 of file PointsMatcher.cpp.

Referenced by OpencvSfM::SequenceAnalyzer::showTracks(), and OpencvSfM::SequenceAnalyzer::showTracksBetween().

4.12.3.7 bool OpencvSfM::PointsMatcher::empty ( ) const [virtual]

Use to know if matching are available

Returns

true if matching has been performed

Definition at line 154 of file PointsMatcher.cpp.

4.12.3.8 const cv::KeyPoint & OpencvSfM::PointsMatcher::getKeypoint (int *numKey*) const

Get a keypoint

Parameters

numKey	index of the wanted point

Returns

keypoint using the cv::Keypoint format

Definition at line 37 of file PointsMatcher.cpp.

4.12.3.9 bool OpencvSfM::PointsMatcher::isMaskSupported ( ) [virtual]

Use this method to know if mask are supported with current matcher

Returns

true if matcher can use mask

Definition at line 86 of file PointsMatcher.cpp.

4.12.3.10 void OpencvSfM::PointsMatcher::knnMatch (cv::Ptr< PointsToTrack > *queryPoints*, std::vector< std::vector< cv::DMatch > > & *matches*, int *k*, const std::vector< cv::Mat > & *masks* = `std::vector<cv::Mat>( )`, bool *compactResult* = `true`) `[virtual]`

Find the k best matches for each descriptor from a query set with train descriptors.

Parameters

queryPoints	Query set of points and descriptors.
matches	Mathes. Each matches[i] is k or less matches for the same query descriptor.
k	Count of best matches will be found per each query descriptor (or less if its not possible).
masks	specifying permissible matches between input query and train matrices of descriptors.
compactResult	Its used when mask (or masks) is not empty. If compactResult is false matches vector will have the same size as queryDescriptors rows. If compactResult is true matches vector will not contain matches for fully masked out query descriptors.

Definition at line 112 of file PointsMatcher.cpp.

4.12.3.11 void OpencvSfM::PointsMatcher::match (cv::Ptr< PointsToTrack > *queryPoints*, std::vector< cv::DMatch > & *matches*, const std::vector< cv::Mat > & *masks* = `std::vector<cv::Mat>( )`) `[virtual]`

Find the k best matches for each descriptor from a query set with train descriptors.

Parameters

queryPoints	Query set of points and descriptors.
matches	Mathes. If some query descriptor (keypoint) masked out in mask no match will be added for this descriptor. So matches size may be less than query keypoints count.
masks	The set of masks. Each masks[i] specifies permissible matches between input query keypoints and stored train keypointss from i-th image.

Definition at line 91 of file PointsMatcher.cpp.

Referenced by crossMatch().

4.12.3.12 void OpencvSfM::PointsMatcher::radiusMatch (cv::Ptr< PointsToTrack > *queryPoints*, std::vector< std::vector< cv::DMatch > > & *matches*, float *maxDistance*, const std::vector< cv::Mat > & *masks* = `std::vector<cv::Mat>( )`, bool *compactResult* = `true`) `[virtual]`

Find the best matches for each query descriptor which have distance less than given threshold.

Parameters

queryPoints	Query set of points and descriptors.
matches	Each matches[i] is k or less matches for the same query descriptor.
maxDistance	The threshold to found match distances.
masks	specifying permissible matches between input query and train matrices of descriptors.
compactResult	Its used when mask (or masks) is not empty. If compactResult is false matches vector will have the same size as queryDescriptors rows. If compactResult is true matches vector will not contain matches for fully masked out query descriptors.

Definition at line 133 of file PointsMatcher.cpp.

4.12.3.13 void OpencvSfM::PointsMatcher::read (const cv::FileNode & *node*, PointsMatcher & *points*) `[static]`

Load the matches from a YAML file.

Parameters

node	Previously opened YAML file node
points	output

Definition at line 267 of file PointsMatcher.cpp.

4.12.3.14 void OpencvSfM::PointsMatcher::train ( ) `[virtual]`

When using matcher which need training, use this method to start the training.

Definition at line 58 of file PointsMatcher.cpp.

Referenced by crossMatch(), knnMatch(), match(), and radiusMatch().

4.12.3.15 void OpencvSfM::PointsMatcher::write (cv::FileStorage & *fs*, const PointsMatcher & *points*) `[static]`

Save the matches into a YAML file.

Parameters

fs	Previously opened YAML file node
points	sequence to save...

Definition at line 292 of file PointsMatcher.cpp.

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

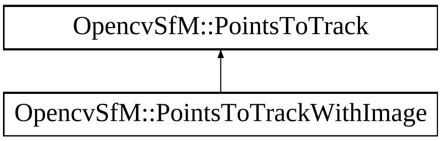
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/PointsMatcher.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/PointsMatcher.cpp

4.13 OpencvSfM::PointsToTrack Class Reference

This class can be used to store informations about point and features. This is an abstract class: you can't use it directly. Use for instance PointsToTrackWithImage.

```
#include <PointsToTrack.h>
```

Inheritance diagram for OpencvSfM::PointsToTrack:



Public Member Functions

- PointsToTrack (int corresponding_image=-1, std::vector< cv::KeyPoint > keypoints=std::vector< cv::KeyPoint >(0), cv::Mat descriptors=cv::Mat())
- virtual ∼PointsToTrack (void)
- void free_descriptors ()
- int computeKeypointsAndDesc (bool forcing_recalculation=false)
- int computeKeypoints ()
- void computeDescriptors ()
- void addKeypoints (std::vector< cv::KeyPoint > keypoints, cv::Mat descriptors=cv::Mat(), bool computeMissingDescriptor=false)
- unsigned int addKeypoint (cv::KeyPoint point)
- const std::vector< cv::KeyPoint > & getKeypoints () const
- void getKeyMatches (const std::vector< TrackOfPoints > &matches, int otherImage, std::vector< cv::Point2f > &pointsVals) const
- const cv::KeyPoint & getKeypoint (unsigned int index) const
- cv::Mat getDescriptors () const

- void printPointsOnImage (const cv::Mat &image, cv::Mat &outImg, const cv::Scalar &color=cv::Scalar::all(-1), int flags=cv::DrawMatchesFlags::DEFAULT) const
- unsigned int getColor (unsigned int index) const

Static Public Member Functions

- static void read (const cv::FileNode &node, PointsToTrack &points)
- static void write (cv::FileStorage &fs, const PointsToTrack &points)

Protected Member Functions

- DECLARE_MUTEX (worker_exclusion)
- virtual int impl_computeKeypoints_ ()
- virtual void impl_computeDescriptors_ ()

Protected Attributes

- unsigned int nb_workers_
- std::vector$<$ cv::KeyPoint $>$ keypoints_
- cv::Mat descriptors_
- std::vector$<$ unsigned int $>$ RGB_values_
- int corresponding_image_

    index of frame when available

Static Protected Attributes

- static int glob_number_images_ = 0

    total numbers of images!

4.13.1 Detailed Description

This class can be used to store informations about point and features. This is an abstract class: you can't use it directly. Use for instance PointsToTrackWithImage.

To create a structure from motion, most methods use points to compute the structure. This class focus on the first task in SfM: find points in image which are easy to track... When available, a feature vector for each points is very helpful: the matching will be easier.

Definition at line 25 of file PointsToTrack.h.

4.13.2 Constructor & Destructor Documentation

4.13.2.1 OpencvSfM::PointsToTrack::PointsToTrack (int *corresponding_image =* `-1`, std::vector< cv::KeyPoint > *keypoints =* `std::vector<cv::KeyPoint>( 0 )`, cv::Mat *descriptors =* `cv::Mat( )`)

this constructor create an object with available information...

Parameters

corresponding-_image	Global index of image
keypoints	the points we will try to track...
descriptors	the feature vector for each points...

Definition at line 15 of file PointsToTrack.cpp.

4.13.2.2 OpencvSfM::PointsToTrack::∼PointsToTrack (void) `[virtual]`

Destructor : delete points and features vectors

Definition at line 42 of file PointsToTrack.cpp.

4.13.3 Member Function Documentation

4.13.3.1 unsigned int OpencvSfM::PointsToTrack::addKeypoint (cv::KeyPoint *point*) `[inline]`

This method is used to add a keypoint at the end of the points vector...

Parameters

point	Keypoints to add

Returns

index of the keypoint.

Definition at line 115 of file PointsToTrack.h.

4.13.3.2 void OpencvSfM::PointsToTrack::addKeypoints (std::vector< cv::KeyPoint > *keypoints,* cv::Mat *descriptors =* `cv::Mat( )`, bool *computeMissingDescriptor =* `false`)

This method is used to add Keypoints...

Parameters

keypoints	Keypoints to add
descriptors	of points, if any
compute-Missing-Descriptor	if true, the missing descriptors are computed.

Definition at line 93 of file PointsToTrack.cpp.

4.13.3.3 void OpencvSfM::PointsToTrack::computeDescriptors ( )

This method is used to compute only descriptors...

Definition at line 86 of file PointsToTrack.cpp.

Referenced by addKeypoints().

4.13.3.4 int OpencvSfM::PointsToTrack::computeKeypoints ( )

This method is used to compute only Keypoints...

Returns

the number of points

Definition at line 77 of file PointsToTrack.cpp.

4.13.3.5 int OpencvSfM::PointsToTrack::computeKeypointsAndDesc (bool *forcing_recalculation* = `false`)

This method is used to compute both Keypoints and descriptors...

Parameters

forcing_-recalculation	if true previous keypoints are removed... If false and if keypoints and descriptor exists, nothing is done.

Returns

the number of points

Definition at line 49 of file PointsToTrack.cpp.

Referenced by write().

4.13.3.6 OpencvSfM::PointsToTrack::DECLARE_MUTEX (worker_exclusion) [protected]

As we want to be able to compute points using parallel execution, and as not every Opencv functions are thread safe, use this mutex to take care of critical portions.

4.13.3.7 void OpencvSfM::PointsToTrack::free_descriptors ( )

To preserve memory, we use this method to free descriptors

Definition at line 29 of file PointsToTrack.cpp.

4.13.3.8 unsigned int OpencvSfM::PointsToTrack::getColor (unsigned int *index*) const [inline]

Use this function to get the color of a point

Parameters

index	of the wanted point

Returns

color packed into the ARGB format

Definition at line 161 of file PointsToTrack.h.

4.13.3.9 cv::Mat OpencvSfM::PointsToTrack::getDescriptors ( ) const [inline]

this method return the descritors for each points in a matrix with size (n$*$m), where n is the number of points and m is the desciptor size.

Returns

descritors for each points in a matrix with size (n$*$m), where n is the number of points and m is the desciptor size.

Definition at line 145 of file PointsToTrack.h.

4.13.3.10 void OpencvSfM::PointsToTrack::getKeyMatches (const std::vector< TrackOfPoints > & *matches,* int *otherImage,* std::vector< cv::Point2f > & *pointsVals*) const

This method update the points coordinates (last parameter) corresponding to tracks containing image index "otherImage"

Parameters

matches	list of tracks. Only points found in tracks are returned
otherImage	index of wanted image
pointsVals	[out] points found in tracks

Definition at line 190 of file PointsToTrack.cpp.

4.13.3.11 const cv::KeyPoint& OpencvSfM::PointsToTrack::getKeypoint (unsigned int *index*) const [inline]

this method return the points coordinates of the i^th entry

Parameters

index	number of keypoints wanted

Returns

points coordinates and when available orientation and size

Definition at line 136 of file PointsToTrack.h.

4.13.3.12 const std::vector<cv::KeyPoint>& OpencvSfM::PointsToTrack::getKeypoints ( ) const [inline]

this method return the points coordinates and sometimes orientation and size

Returns

points coordinates and when available orientation and size

Definition at line 121 of file PointsToTrack.h.

4.13.3.13 virtual void OpencvSfM::PointsToTrack::impl_computeDescriptors_ ( ) [inline, protected, virtual]

This is the method you should implement when you create a new descriptors extractor...

Reimplemented in OpencvSfM::PointsToTrackWithImage.

Definition at line 67 of file PointsToTrack.h.

Referenced by computeDescriptors(), and computeKeypointsAndDesc().

4.13.3.14 virtual int OpencvSfM::PointsToTrack::impl_computeKeypoints_ ( ) [inline, protected, virtual]

This is the method you should implement when you create a new point detector algorithm.

Returns

the number of points

Reimplemented in OpencvSfM::PointsToTrackWithImage.

Definition at line 62 of file PointsToTrack.h.

Referenced by computeKeypoints(), and computeKeypointsAndDesc().

4.13.3.15 void OpencvSfM::PointsToTrack::printPointsOnImage (const cv::Mat & *image*, cv::Mat & *outImg*, const cv::Scalar & *color* = `cv::Scalar::all( -1 )`, int *flags* = `cv::DrawMatchesFlags::DEFAULT`) const

To show the points on image, use this function to draw points on it.

Parameters

image	Source image.
outImg	Output image. Its content depends on flags value what is drawn in output image. See possible flags bit values.
color	Color of keypoints
flags	Possible flags bit values is defined by DrawMatchesFlags (see `http://opencv.willowgarage.com/documentation/cpp/features2d_drawing_function_of_keypoints_and_matches.html#cv-drawmatches`)

Definition at line 123 of file PointsToTrack.cpp.

4.13.3.16 void OpencvSfM::PointsToTrack::read (const cv::FileNode & *node*, PointsToTrack & *points*) `[static]`

Load the points from a YAML file.

Parameters

node	Previously opened YAML file node
points	output

Definition at line 129 of file PointsToTrack.cpp.

4.13.3.17 void OpencvSfM::PointsToTrack::write (cv::FileStorage & *fs*, const PointsToTrack & *points*) `[static]`

Save the points into a YAML file.

Parameters

fs	Previously opened YAML file node
points	sequence to save...

Definition at line 171 of file PointsToTrack.cpp.

4.13.4 Member Data Documentation

4.13.4.1 cv::Mat **OpencvSfM::PointsToTrack::descriptors_** [protected]

this attribute will store descritors for each points in a matrix with size (n∗m), where n is the number of points and m is the desciptor size.

Definition at line 49 of file PointsToTrack.h.

Referenced by addKeypoints(), computeKeypointsAndDesc(), free_descriptors(), -OpencvSfM::PointsToTrackWithImage::impl_computeDescriptors_(), read(), write(), and ∼PointsToTrack().

4.13.4.2 std::vector<cv::KeyPoint> **OpencvSfM::PointsToTrack::keypoints_** [protected]

This attribute will store points coordinates and sometimes orientation and size

Definition at line 43 of file PointsToTrack.h.

Referenced by addKeypoints(), computeKeypoints(), computeKeypointsAndDesc(), OpencvSfM::PointsToTrackWithImage::getColorOfPoints(), getKeyMatches(), OpencvSfM::PointsToTrackWithImage::impl_computeDescriptors_(), OpencvSfM::PointsToTrackWithImage::impl_computeKeypoints_(), printPointsOnImage(), read(), write(), and ∼PointsToTrack().

4.13.4.3 unsigned int **OpencvSfM::PointsToTrack::nb_workers_** [protected]

To preserve memory, we need to know how many process are working with theses points...

Definition at line 38 of file PointsToTrack.h.

Referenced by computeKeypointsAndDesc(), free_descriptors(), OpencvSfM::PointsToTrackWithImage::impl_computeDescriptors_(), PointsToTrack(), and read().

4.13.4.4 std::vector<unsigned int> **OpencvSfM::PointsToTrack::RGB_values_** [protected]

When available, the color of each point can be stored here.

Definition at line 53 of file PointsToTrack.h.

Referenced by OpencvSfM::PointsToTrackWithImage::getColorOfPoints(), read(), and write().

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

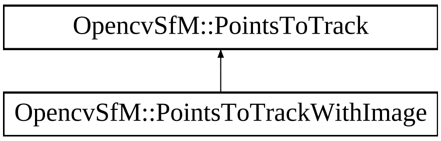
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/PointsToTrack.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/PointsToTrack.cpp

4.14 OpencvSfM::PointsToTrackWithImage Class Reference

This class can be used to find points and features in pictures using SIFT detector.

```
#include <PointsToTrackWithImage.h>
```

Inheritance diagram for OpencvSfM::PointsToTrackWithImage:



Public Member Functions

- PointsToTrackWithImage (int corresponding_image, cv::Mat imageToAnalyse, cv::Mat maskOfAnalyse, cv::Ptr< cv::FeatureDetector > feature_detector=0, cv::Ptr< cv::DescriptorExtractor > descriptor_detector=0)
- PointsToTrackWithImage (int corresponding_image, cv::Mat imageToAnalyse, cv::Mat maskOfAnalyse, std::string feature_detector, std::string descriptor_detector="SIFT")
- void setFeatureDetector (cv::Ptr< cv::FeatureDetector > feature_detector)
- void setDescriptorExtractor (cv::Ptr< cv::DescriptorExtractor > descriptor_detector)
- void getColorOfPoints ()
- cv::Mat getImage ()

Protected Member Functions

- int impl_computeKeypoints_ ()
- void impl_computeDescriptors_ ()

Protected Attributes

- cv::Ptr< cv::FeatureDetector > feature_detector_ *class which will find the points*

- cv::Ptr< cv::DescriptorExtractor > descriptor_detector_

  class which will compute the descriptors
- cv::Mat imageToAnalyse_

  Picture from where points are detected.
- cv::Mat maskOfAnalyse_

  Mask of analyse. Everything out of this mask is ignored.

4.14.1 Detailed Description

This class can be used to find points and features in pictures using SIFT detector.

To create a structure from motion, most methods use points to compute the structure. This class focus on the first task in SfM: find points in image which are easy to track... When available, a feature vector for each points is very helpful: the matching will be easier.

Definition at line 15 of file PointsToTrackWithImage.h.

4.14.2 Constructor & Destructor Documentation

4.14.2.1 OpencvSfM::PointsToTrackWithImage::PointsToTrackWithImage (int *corresponding_image*, cv::Mat *imageToAnalyse*, cv::Mat *maskOfAnalyse*, cv::Ptr< cv::FeatureDetector > *feature_detector* = 0, cv::Ptr< cv::DescriptorExtractor > *descriptor_detector* = 0)

First constructor used to create a list of points to track using a feature and a descriptor algorithm.

Parameters

corresponding_image	Global index of image
imageToAnalyse	Image to use for keypoints and features search
maskOfAnalyse	Mask used to hide part of image
feature_detector	Algorithm to use for features detection (see `http://opencv.willowgarage.com/documentation/cpp/common_interfaces_for_feature_detection_and_descriptor_extraction.html#featuredetector`)
descriptor_detector	Algorithm to use for descriptors detection (see `http://opencv.willowgarage.com/documentation/cpp/common_interfaces_for_feature_detection_and_descriptor_extraction.html#descriptorextractor`)

Definition at line 20 of file PointsToTrackWithImage.cpp.

4.14.2.2 OpencvSfM::PointsToTrackWithImage::PointsToTrackWithImage (int *corresponding_image,* cv::Mat *imageToAnalyse,* cv::Mat *maskOfAnalyse,* std::string *feature_detector,* std::string *descriptor_detector* = "SIFT")

Second constructor used to create a list of points to track using a feature and a descriptor algorithm.

Parameters

corresponding_image	Global index of image
imageToAnalyse	Image to use for keypoints and features search
maskOfAnalyse	Mask used to hide part of image
feature_detector	name of the algorithm to use for features detection (see http://opencv.willowgarage.com/documentation/cpp/common_interfaces_for_feature_detection_and_descriptor_extraction.html#featuredetector)
descriptor_detector	name of the algorithm to use for descriptors detection (see http://opencv.willowgarage.com/documentation/cpp/common_interfaces_for_feature_detection_and_descriptor_extraction.html#descriptorextractor)

Definition at line 30 of file PointsToTrackWithImage.cpp.

4.14.3 Member Function Documentation

4.14.3.1 void OpencvSfM::PointsToTrackWithImage::getColorOfPoints ( )

This method is used to get color for each points...

Definition at line 56 of file PointsToTrackWithImage.cpp.

Referenced by impl_computeDescriptors_(), and impl_computeKeypoints_().

4.14.3.2 cv::Mat OpencvSfM::PointsToTrackWithImage::getImage ( ) [inline]

Get the image used to compute points

Definition at line 76 of file PointsToTrackWithImage.h.

4.14.3.3 void OpencvSfM::PointsToTrackWithImage::impl_computeDescriptors_ ( ) [protected, virtual]

This method is used to compute only descriptors...

Reimplemented from OpencvSfM::PointsToTrack.

Definition at line 106 of file PointsToTrackWithImage.cpp.

4.14.3.4 int OpencvSfM::PointsToTrackWithImage::impl_computeKeypoints_ ( ) [protected, virtual]

This method is used to compute only Keypoints...

Returns

the number of points

Reimplemented from OpencvSfM::PointsToTrack.

Definition at line 98 of file PointsToTrackWithImage.cpp.

4.14.3.5 void OpencvSfM::PointsToTrackWithImage::setDescriptorExtractor (cv::Ptr< cv::DescriptorExtractor > *descriptor_detector*)

Use this function to set the descriptor extractor. Can be useful to update parameters, for example!

Parameters

descriptor_detector	new pointer of a descriptor extractor algorithm.

Definition at line 45 of file PointsToTrackWithImage.cpp.

4.14.3.6 void OpencvSfM::PointsToTrackWithImage::setFeatureDetector (cv::Ptr< cv::FeatureDetector > *feature_detector*)

Use this function to set the feature detector. Can be useful to update parameters, for example!

Parameters

feature_detector	new pointer of a feature detector algorithm.

Definition at line 40 of file PointsToTrackWithImage.cpp.

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

- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/PointsToTrackWithImage.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/PointsToTrackWithImage.cpp

4.15 OpencvSfM::SequenceAnalyzer Class Reference

This class tries to match points in the entire sequence. It follow ideas proposed by Noah Snavely: Modeling the World from Internet Photo Collections.

```
#include <SequenceAnalyzer.h>
```

Public Member Functions

- SequenceAnalyzer (MotionProcessor input_sequence, cv::Ptr< cv::FeatureDetector > feature_detector, cv::Ptr< cv::DescriptorExtractor > descriptor_extractor, cv::Ptr< PointsMatcher > match_algorithm)
- SequenceAnalyzer (std::vector< cv::Ptr< PointsToTrack > > &points_to_track, cv::Ptr< PointsMatcher > match_algorithm, const std::vector< cv::Mat > &images)
- SequenceAnalyzer (cv::FileNode file, std::vector< cv::Mat > &images=std::vector< cv::Mat >())
- ∼SequenceAnalyzer (void)
- void addNewImage (cv::Mat image, cv::Ptr< PointsToTrack > points=cv::Ptr< PointsToTrack >())
- void computeMatches ()
- std::vector< TrackOfPoints > & getTracks ()
- std::vector< cv::Ptr < PointsToTrack > > & getPoints ()
- ImagesGraphConnection & getImgGraph ()
- void showTracks (int timeBetweenImg=25)
- void showTracks (int img_to_show, int timeBetweenImg)
- void showTracksBetween (unsigned int img1, unsigned int img2)
- int getNumViews () const
- cv::Mat getImage (int idx)
- void addMatches (std::vector< cv::DMatch > &newMatches, unsigned int img1, unsigned int img2)
- void addTracks (std::vector< TrackOfPoints > &newTracks)
- void constructImagesGraph ()
- std::vector< unsigned int > getColors ()
- std::vector< cv::Vec3d > get3DStructure ()
- void showPointsOnImage (unsigned int i, const std::vector< cv::Vec2d > &pixelProjection)
- std::vector< cv::Ptr < PointsToTrack > > getPointsToTrack ()

Static Public Member Functions

- static void keepOnlyCorrectMatches (std::vector< TrackOfPoints > &tracks, unsigned int min_matches=10, unsigned int min_consistance=3)
- static void keepOnlyCorrectMatches (SequenceAnalyzer &tracks, unsigned int min_matches=10, unsigned int min_consistance=3)
- static void read (const cv::FileNode &node, SequenceAnalyzer &points)
- static void write (cv::FileStorage &fs, const SequenceAnalyzer &points)

Protected Attributes

- cv::Ptr< cv::FeatureDetector > feature_detector_
- cv::Ptr< cv::DescriptorExtractor > descriptor_extractor_
- std::vector< cv::Ptr < PointsToTrack > > points_to_track_
- std::vector< cv::Mat > images_
- cv::Ptr< PointsMatcher > match_algorithm_
- std::vector< cv::Ptr < PointsMatcher > > matches_
- std::vector< TrackOfPoints > tracks_
- ImagesGraphConnection images_graph_

Static Protected Attributes

- static int mininum_points_matches = 20

  Minimum points detected into an image to keep this estimation (set to 20)
- static int mininum_image_matches = 2

  Minimum images connections in a track to keep this estimation (usually set to 2)

4.15.1 Detailed Description

This class tries to match points in the entire sequence. It follow ideas proposed by Noah Snavely: Modeling the World from Internet Photo Collections.

This class process an input video to first extracts the features, then matches them and keeps them only when there is more than 2 pictures containing the point.

Definition at line 25 of file SequenceAnalyzer.h.

4.15.2 Constructor & Destructor Documentation

4.15.2.1 OpencvSfM::SequenceAnalyzer::SequenceAnalyzer (MotionProcessor *input_sequence*, cv::Ptr< cv::FeatureDetector > *feature_detector*, cv::Ptr< cv::DescriptorExtractor > *descriptor_extractor*, cv::Ptr< PointsMatcher > *match_algorithm*)

Constructor taking a MotionProcessor to load images and a features detector and descriptor to find matches.

Parameters

input_-sequence	input images
feature_-detector	Algorithm to use for features detection (see http://opencv.willowgarage.com/documentation/cpp/common_interfaces_for_feature_detection_and_descriptor_extraction.html#featuredetector)
descriptor_-extractor	Algorithm to use for descriptors detection (see http://opencv.willowgarage.com/documentation/cpp/common_interfaces_for_feature_detection_and_descriptor_extraction.html#descriptorextractor)
match_-algorithm	algorithm to match points of each images

Definition at line 25 of file SequenceAnalyzer.cpp.

4.15.2.2 OpencvSfM::SequenceAnalyzer::SequenceAnalyzer (std::vector< cv::Ptr< PointsToTrack > > & *points_to_track*, cv::Ptr< PointsMatcher > *match_algorithm*, const std::vector< cv::Mat > & *images*)

Constructor taking a vector of points to track and a PointsMatcher algorithm to find matches.

Parameters

images	input images. Points should be in the same order!
points_to_-track	list of points to track with (or not) features
match_-algorithm	algorithm to match points of each images

Definition at line 56 of file SequenceAnalyzer.cpp.

4.15.2.3 OpencvSfM::SequenceAnalyzer::SequenceAnalyzer (cv::FileNode *file*, std::vector< cv::Mat > & *images* = std::vector<cv::Mat>())

Constructor taking a list of images and a FileNode

Parameters

images	input images. Points should be in the same order!
file	YAML file to get points and matches

Definition at line 68 of file SequenceAnalyzer.cpp.

4.15.2.4 OpencvSfM::SequenceAnalyzer::∼SequenceAnalyzer (void)

Destructor of SequenceAnalyzer (nothing is released!)

Definition at line 75 of file SequenceAnalyzer.cpp.

4.15.3 Member Function Documentation

4.15.3.1 void OpencvSfM::SequenceAnalyzer::addMatches (std::vector< cv::DMatch > & *newMatches,* unsigned int *img1,* unsigned int *img2*) [inline]

This function add matches to tracks

Parameters

newMatches	new matches to add
img1	index of source matches image
img2	index of destination matches image

Definition at line 181 of file SequenceAnalyzer.cpp.

4.15.3.2 void OpencvSfM::SequenceAnalyzer::addNewImage (cv::Mat *image,* cv::Ptr< PointsToTrack > *points* = cv::Ptr<PointsToTrack>( ))

This method add new image to track. When adding, if the matches are not computed, use automatically computeMatches to compute them!

Parameters

image	New image
points	extracted points with features vectors.

Definition at line 79 of file SequenceAnalyzer.cpp.

4.15.3.3 void OpencvSfM::SequenceAnalyzer::addTracks (std::vector< TrackOfPoints > & *newTracks*)

This function add new Tracks

Parameters

newTracks	new Tracks to add

Definition at line 227 of file SequenceAnalyzer.cpp.

4.15.3.4 void OpencvSfM::SequenceAnalyzer::computeMatches ( )

This method compute the matches between each points of each images. It first compute missing features descriptor, then train each matcher. Finally compute tracks of keypoints (a track is a connected set of matching keypoints across multiple images)

Definition at line 98 of file SequenceAnalyzer.cpp.

4.15.3.5 void OpencvSfM::SequenceAnalyzer::constructImagesGraph ( )

This function constructs and feeds the images_graph_

Definition at line 524 of file SequenceAnalyzer.cpp.

4.15.3.6 std::vector< cv::Vec3d > OpencvSfM::SequenceAnalyzer::get3DStructure ( )

This function will create a list of 3D points corresponding to object viewed in the sequence

Definition at line 548 of file SequenceAnalyzer.cpp.

4.15.3.7 std::vector< unsigned int > OpencvSfM::SequenceAnalyzer::getColors ( )

This function will create a list of points color corresponding to object viewed in the sequence

Definition at line 561 of file SequenceAnalyzer.cpp.

4.15.3.8 cv::Mat OpencvSfM::SequenceAnalyzer::getImage (int *idx*) [inline]

get the ith image. No checks are performed!

Parameters

idx	index of the wanted image

Returns

Matrix of the wanted image

Definition at line 209 of file SequenceAnalyzer.h.

4.15.3.9 ImagesGraphConnection& OpencvSfM::SequenceAnalyzer::getImgGraph ( ) [inline]

Get the graph of image connections

Returns

graph of image connections

Definition at line 148 of file SequenceAnalyzer.h.

Referenced by OpencvSfM::EuclideanEstimator::computeReconstruction().

4.15.3.10 int OpencvSfM::SequenceAnalyzer::getNumViews ( ) const [inline]

Use this function to know how many images are stored into tracks...

Returns

numbers of images (and cameras) stored into tracks.

Definition at line 188 of file SequenceAnalyzer.h.

4.15.3.11 std::vector< cv::Ptr< PointsToTrack > >& OpencvSfM::SequenceAnalyzer::getPoints ( ) [inline]

This method can be used to get the points

Definition at line 141 of file SequenceAnalyzer.h.

Referenced by OpencvSfM::EuclideanEstimator::bundleAdjustement(), OpencvSfM::EuclideanEstimator::cameraResection(), OpencvSfM::EuclideanEstimator::computeReconstruction(), OpencvSfM::StructureEstimator::computeStructure(), OpencvSfM::EuclideanEstimator::initialReconstruction(), and OpencvSfM::StructureEstimator::removeOutliersTracks().

4.15.3.12 std::vector< cv::Ptr< PointsToTrack > > OpencvSfM::SequenceAnalyzer::getPointsToTrack ( ) [inline]

Get the points for track from this sequence.

Returns

points for track from this sequence.

Definition at line 250 of file SequenceAnalyzer.h.

4.15.3.13 std::vector<TrackOfPoints>& OpencvSfM::SequenceAnalyzer::getTracks ( ) [inline]

This method can be used to get the tracks

Definition at line 137 of file SequenceAnalyzer.h.

Referenced by OpencvSfM::EuclideanEstimator::computeReconstruction(), OpencvSfM::StructureEstimator::computeStructure(), OpencvSfM::EuclideanEstimator::initialReconstruction(), keepOnlyCorrectMatches(), and OpencvSfM::StructureEstimator::removeOutliersTracks().

4.15.3.14 void OpencvSfM::SequenceAnalyzer::keepOnlyCorrectMatches (std::vector< TrackOfPoints > & *tracks*, unsigned int *min_matches* = 10, unsigned int *min_consistance* = 3) [static]

This method keep only tracks with more than mininum_image_matches

Definition at line 158 of file SequenceAnalyzer.cpp.

Referenced by OpencvSfM::EuclideanEstimator::computeReconstruction().

4.15.3.15 static void OpencvSfM::SequenceAnalyzer::keepOnlyCorrectMatches (SequenceAnalyzer & *tracks*, unsigned int *min_matches* = 10, unsigned int *min_consistance* = 3) [inline, static]

This method keep only tracks with more than mininum_image_matches

Definition at line 127 of file SequenceAnalyzer.h.

4.15.3.16 void OpencvSfM::SequenceAnalyzer::read (const cv::FileNode & *node*, SequenceAnalyzer & *points*) [static]

Load the sequence from a YAML file.

Parameters

node	Previously opened YAML file node
points	output

Definition at line 384 of file SequenceAnalyzer.cpp.

Referenced by SequenceAnalyzer().

4.15.3.17 void OpencvSfM::SequenceAnalyzer::showPointsOnImage (unsigned int *i*, const std::vector< cv::Vec2d > & *pixelProjection*)

Use this function to show 2D points into ith image

Parameters

i	index of wanted image
pixelProjection	list of 2D points

Definition at line 574 of file SequenceAnalyzer.cpp.

4.15.3.18 void OpencvSfM::SequenceAnalyzer::showTracks (int *timeBetweenImg =* `25`)

Use this function to print the sequence of matches

Parameters

timeBetweenImg	see cv::waitKey for the value

Definition at line 241 of file SequenceAnalyzer.cpp.

4.15.3.19 void OpencvSfM::SequenceAnalyzer::showTracks (int *img_to_show,* int *timeBetweenImg*)

Use this function to print the sequence of matches

Parameters

img_to_show	index of image whose tracks will be shown.
timeBetweenImg	see cv::waitKey for the value

Definition at line 293 of file SequenceAnalyzer.cpp.

4.15.3.20 void OpencvSfM::SequenceAnalyzer::showTracksBetween (unsigned int *img1,* unsigned int *img2*)

Use this function to print the matches between two images

Definition at line 342 of file SequenceAnalyzer.cpp.

4.15.3.21 void OpencvSfM::SequenceAnalyzer::write (cv::FileStorage & *fs,* const SequenceAnalyzer & *points*) `[static]`

Save the sequence into a YAML file.

Parameters

fs	Previously opened YAML file node
points	sequence to save...

Definition at line 467 of file SequenceAnalyzer.cpp.

4.15.4 Member Data Documentation

4.15.4.1 cv::Ptr<cv::DescriptorExtractor> OpencvSfM::SequenceAnalyzer::descriptor_extractor_ [protected]

optional, method to use for feature extraction

Definition at line 37 of file SequenceAnalyzer.h.

Referenced by addNewImage().

4.15.4.2 cv::Ptr<cv::FeatureDetector> OpencvSfM::SequenceAnalyzer::feature_detector_ [protected]

optional, method to use for feature detection

Definition at line 33 of file SequenceAnalyzer.h.

Referenced by addNewImage().

4.15.4.3 std::vector<cv::Mat> OpencvSfM::SequenceAnalyzer::images_ [protected]

List of input images

Definition at line 45 of file SequenceAnalyzer.h.

Referenced by addNewImage(), SequenceAnalyzer(), showPointsOnImage(), showTracks(), and showTracksBetween().

4.15.4.4 ImagesGraphConnection OpencvSfM::SequenceAnalyzer::images_graph_ [protected]

Graph of images relations (value (i,j) correspond to the numbers of matches between theses two images

Definition at line 64 of file SequenceAnalyzer.h.

Referenced by constructImagesGraph().

4.15.4.5 cv::Ptr<PointsMatcher> OpencvSfM::SequenceAnalyzer::match_algorithm_ [protected]

The matcher algorithm we should use to find matches.

Definition at line 49 of file SequenceAnalyzer.h.

Referenced by computeMatches(), and read().

4.15.4.6 std::vector< cv::Ptr< PointsMatcher > > OpencvSfM::SequenceAnalyzer::matches_ [protected]

A matcher for each picture. Its role is to find quickly matches between i^th picture and other images.

Definition at line 54 of file SequenceAnalyzer.h.

Referenced by read().

4.15.4.7 std::vector< cv::Ptr< PointsToTrack > > OpencvSfM::SequenceAnalyzer::points_to_track_ [protected]

A list of points for each picture

Definition at line 41 of file SequenceAnalyzer.h.

Referenced by addNewImage(), computeMatches(), constructImagesGraph(), read(), SequenceAnalyzer(), showTracks(), showTracksBetween(), and write().

4.15.4.8 std::vector<TrackOfPoints> OpencvSfM::SequenceAnalyzer::tracks_ [protected]

List of each tracks found. A track is a connected set of matching keypoints across multiple images

Definition at line 59 of file SequenceAnalyzer.h.

Referenced by addMatches(), addTracks(), computeMatches(), constructImagesGraph(), get3DStructure(), getColors(), read(), showTracks(), showTracksBetween(), and write().

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

- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/SequenceAnalyzer.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/SequenceAnalyzer.cpp

4.16 OpencvSfM::StructureEstimator Class Reference

This class tries to find the 3D structure using a sequence and cameras fully parameterized.

```
#include <StructureEstimator.h>
```

Public Member Functions

- StructureEstimator (SequenceAnalyzer *sequence, std::vector< PointOfView > *cameras, int max_repro_error=10)

- ∼StructureEstimator ()
- std::vector< char > computeStructure (unsigned int max_error=10)
- std::vector< TrackOfPoints > computeStructure (const std::vector< int > &list_of_images, unsigned int max_error=10)
- void removeOutliersTracks (double max_error=10, std::vector< TrackOfPoints > ∗list_of_tracks=NULL)

Protected Attributes

- SequenceAnalyzer ∗ sequence_

  Object containing all 2D information of this sequence.
- std::vector< PointOfView > ∗ cameras_

  List of cameras (intra and extern parameters...)
- int max_repro_error_

  Maximum reprojection error allowed.

4.16.1 Detailed Description

This class tries to find the 3D structure using a sequence and cameras fully parameterized.

Definition at line 16 of file StructureEstimator.h.

4.16.2 Constructor & Destructor Documentation

4.16.2.1 OpencvSfM::StructureEstimator::StructureEstimator (SequenceAnalyzer ∗ *sequence*, std::vector< PointOfView > ∗ *cameras*, int *max_repro_error* = `10`) `[inline]`

Constructor of this 3D structure estimator

Parameters

sequence	the address of the object containing all 2D information of this sequence
cameras	List of cameras (intra and extern parameters...)
max_repro_error	Maximum reprojection error allowed

Definition at line 30 of file StructureEstimator.h.

4.16.2.2 OpencvSfM::StructureEstimator::∼StructureEstimator ( ) `[inline]`

Destructor will not release datas as they where given by address!

Definition at line 38 of file StructureEstimator.h.

4.16.3 Member Function Documentation

4.16.3.1 vector< char > OpencvSfM::StructureEstimator::computeStructure (unsigned int *max_error* = 10)

Project previously 2D points matches using cameras parameters

Parameters

max_error	maximum error allowed.

Returns

the mask of correct points (0 if error > max_error)

Definition at line 12 of file StructureEstimator.cpp.

Referenced by OpencvSfM::EuclideanEstimator::computeReconstruction(), and OpencvSfM::EuclideanEstimator::initialReconstruction().

4.16.3.2 std::vector< TrackOfPoints > OpencvSfM::StructureEstimator::computeStructure (const std::vector< int > & *list_of_images*, unsigned int *max_error* = 10)

Project previously 2D points matches for only two views

Parameters

list_of_images	list of image indexes to use
max_error	maximum error allowed.

Returns

output of tracks triangulated (contain 3D point)

Definition at line 46 of file StructureEstimator.cpp.

4.16.3.3 void OpencvSfM::StructureEstimator::removeOutliersTracks (double *max_error* = 10, std::vector< TrackOfPoints > * *list_of_tracks* = NULL)

Remove points from track when projection error > max_error

Parameters

max_error	maximum error of back projection allowed
list_of_tracks	list of tracks to work with. If NULL or not set, will use StructureEstimator::sequence_

Definition at line 95 of file StructureEstimator.cpp.

Referenced by OpencvSfM::EuclideanEstimator::computeReconstruction().

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

- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/-StructureEstimator.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/-StructureEstimator.cpp

4.17 OpencvSfM::TrackOfPoints Class Reference

This class store the track of keypoints. A track is a connected set of matching keypoints across multiple images.

```
#include <TracksOfPoints.h>
```

Public Member Functions

- template<typename Type , int size>
  operator cv::Vec< Type, size > & ()
- bool addMatch (const int image_src, const int point_idx)
- bool containImage (const int image_wanted) const
- bool containPoint (const int image_src, const int point_idx1) const
- unsigned int getNbTrack () const
- cv::DMatch toDMatch (const int img1, const int img2) const
- void getMatch (const unsigned int index, int &idImage, int &idPoint) const
- int getPointIndex (const unsigned int image) const
- int getImageIndex (const unsigned int idx) const
- double triangulateLinear (std::vector< PointOfView > &cameras, const std::vector< cv::Ptr< PointsToTrack > > &points_to_track, cv::Vec3d &points3D, const std::vector< bool > &masks=std::vector< bool >())
- double triangulateRobust (std::vector< PointOfView > &cameras, const std::vector< cv::Ptr< PointsToTrack > > &points_to_track, cv::Vec3d &points3D, double reproj_error=4, const std::vector< bool > &masks=std::vector< bool >())
- void removeOutliers (std::vector< PointOfView > &cameras, const std::vector< cv::Ptr< PointsToTrack > > &points_to_track, double reproj_error=4, std::vector< bool > *masks=NULL)
- void set3DPosition (cv::Vec3d newPoint)
- cv::Ptr< cv::Vec3d > get3DPosition ()
- unsigned int getColor () const
- void setColor (unsigned int c)

Static Public Member Functions

- static void keepTrackHavingImage (unsigned int idx_image, std::vector< TrackOfPoints > &tracks)
- static void keepTrackWithImages (const std::vector< int > &imgList, std::vector< TrackOfPoints > &tracks)
- static void mixTracks (const std::vector< TrackOfPoints > &list_tracks, std::vector< TrackOfPoints > ∗mixed_tracks)

Protected Member Functions

- double errorEstimate (std::vector< PointOfView > &cameras, const std::vector< cv::Ptr< PointsToTrack > > &points_to_track, cv::Vec3d &points3D, const std::vector< bool > &masks=std::vector< bool >()) const

Protected Attributes

- cv::Ptr< cv::Vec3d > point3D

  The corresponding 3D coordinates. If not available, Ptr is empty.
- std::vector< unsigned int > images_indexes_

  List of image indexes of unordered points.
- std::vector< unsigned int > point_indexes_

  List of point indexes of unordered points.
- unsigned int color

  Color of this point (computed using the mean of every 2D points projections.
- std::vector< bool > good_values
- int track_consistance

Friends

- class **SequenceAnalyzer**

4.17.1 Detailed Description

This class store the track of keypoints. A track is a connected set of matching keypoints across multiple images.

This class can be used as a Vec3d because it's the projection of a 3D points Of course, use triangulate method before to create this 3D point!

Discussion: Store index of points or 2D position?

Definition at line 23 of file TracksOfPoints.h.

4.17.2 Member Function Documentation

4.17.2.1 bool OpencvSfM::TrackOfPoints::addMatch (const int *image_src,* const int *point_idx*)

This function add matches to track

Parameters

image_src	index of source matches image
point_idx	index of point in source image

Returns

true if this match is correct, false if inconsistent with Snavely's rules.

Definition at line 56 of file TracksOfPoints.cpp.

Referenced by OpencvSfM::SequenceAnalyzer::addMatches(), mixTracks(), and -OpencvSfM::SequenceAnalyzer::read().

4.17.2.2 bool OpencvSfM::TrackOfPoints::containImage (const int *image_wanted*) const [inline]

This function is used to know if the track contains the image

Parameters

image_-wanted	index of query image

Returns

true if this track contain points from the query image

Definition at line 66 of file TracksOfPoints.h.

Referenced by OpencvSfM::StructureEstimator::computeStructure(), OpencvSfM::-PointsToTrack::getKeyMatches(), OpencvSfM::EuclideanEstimator::initialReconstruction(), keepTrackHavingImage(), and keepTrackWithImages().

4.17.2.3 bool OpencvSfM::TrackOfPoints::containPoint (const int *image_src,* const int *point_idx1*) const

This function is used to know if the track contains the query point

Parameters

image_src	index of query image
point_idx1	index of point in query image

Returns

true if this track contain the point from the query image

Definition at line 95 of file TracksOfPoints.cpp.

Referenced by OpencvSfM::SequenceAnalyzer::addMatches(), and mixTracks().

4.17.2.4 double OpencvSfM::TrackOfPoints::errorEstimate (std::vector< PointOfView > & *cameras*, const std::vector< cv::Ptr< PointsToTrack > > & *points_to_track*, cv::Vec3d & *points3D*, const std::vector< bool > & *masks* = `std::vector<bool>( )`) const `[protected]`

Comptue an estimation of 2D reprojection error

Parameters

cameras	list of cameras
points_to_-track	list of 2D points
points3D	3d points to project with cameras
masks	wanted points

Returns

estimation of distance detween projections and measures.

Definition at line 156 of file TracksOfPoints.cpp.

Referenced by triangulateLinear().

4.17.2.5 cv::Ptr<cv::Vec3d> OpencvSfM::TrackOfPoints::get3DPosition ( ) `[inline]`

Use this function to get the 3D point corresponding to this track

Returns

pointer on the 3D coordinates (could be NULL!)

Definition at line 183 of file TracksOfPoints.h.

Referenced by OpencvSfM::StructureEstimator::removeOutliersTracks().

4.17.2.6 unsigned int OpencvSfM::TrackOfPoints::getColor ( ) const `[inline]`

Use this function to get the color of this track

Returns

color of this track (ARGB packed into a int)

Definition at line 189 of file TracksOfPoints.h.

Referenced by OpencvSfM::SequenceAnalyzer::write().

4.17.2.7 int OpencvSfM::TrackOfPoints::getImageIndex (const unsigned int *idx*) const [inline]

use this function to get the image corresponding to the nth entry of this track

Parameters

idx	index of wanted point

Returns

number of image

Definition at line 124 of file TracksOfPoints.h.

Referenced by OpencvSfM::StructureEstimator::computeStructure().

4.17.2.8 void OpencvSfM::TrackOfPoints::getMatch (const unsigned int *index,* int & *idImage,* int & *idPoint*) const

use this function to get the n^th match value from this track

Parameters

index	which match
idImage	out value of the image index
idPoint	out value of the point index

Definition at line 145 of file TracksOfPoints.cpp.

4.17.2.9 unsigned int OpencvSfM::TrackOfPoints::getNbTrack ( ) const [inline]

This function is used to get the numbers of image for this track

Returns

0 if inconsistent, >= 2 else

Definition at line 85 of file TracksOfPoints.h.

Referenced by OpencvSfM::StructureEstimator::computeStructure(), OpencvSfM::StructureEstimator::removeOutliersTracks(), and OpencvSfM::SequenceAnalyzer-

::write().

4.17.2.10 int OpencvSfM::TrackOfPoints::getPointIndex (const unsigned int *image*) const [inline]

use this function to get the index point of the wanted image

Parameters

image	index of wanted image

Returns

index of point

Definition at line 110 of file TracksOfPoints.h.

Referenced by OpencvSfM::PointsToTrack::getKeyMatches().

4.17.2.11 void OpencvSfM::TrackOfPoints::keepTrackHavingImage (unsigned int *idx_image*, std::vector< **TrackOfPoints** > & *tracks*) [static]

Use this function to keep only tracks having image from first parameter

Parameters

idx_image	index of needed image
tracks	vector of matches to clean...

Definition at line 341 of file TracksOfPoints.cpp.

4.17.2.12 void OpencvSfM::TrackOfPoints::keepTrackWithImages (const std::vector< int > & *imgList*, std::vector< **TrackOfPoints** > & *tracks*) [static]

Use this function to keep only tracks having at least 2 images from first parameter

Parameters

imgList	Needed images indexes
tracks	vector of matches to clean...

Definition at line 406 of file TracksOfPoints.cpp.

4.17.2.13 void OpencvSfM::TrackOfPoints::mixTracks (const std::vector< **TrackOfPoints** > & *list_tracks,* std::vector< **TrackOfPoints** > * *mixed_tracks*) [static]

add to mixed_tracks the new tracks from list_tracks who are not in mixed_tracks. Of course, as a track of points contains only indexes, be careful to mix two compatible vectors (i.e. share the same points indexes)

Parameters

list_tracks	first list of tracks to add into mixed_tracks
mixed_-tracks	output list of tracks

Definition at line 361 of file TracksOfPoints.cpp.

4.17.2.14 template<typename Type , int size> OpencvSfM::TrackOfPoints::operator cv::Vec< Type, size > & ( ) [inline]

cast operator to use this object as a 3D point!

Definition at line 47 of file TracksOfPoints.h.

4.17.2.15 void OpencvSfM::TrackOfPoints::removeOutliers (std::vector< **PointOfView** > & *cameras,* const std::vector< cv::Ptr< **PointsToTrack** > > & *points_to_track,* double *reproj_error =* 4*,* std::vector< bool > * *masks =* NULL)

From the list of points of this track, remove each 2D points when reprojection error > reproj_error

Parameters

cameras	cameras used to compute projection of 3D point.
points_to_-track	2D points used to compute reprojection error
reproj_error	Threshold used to reject outliners
masks	used to knwo which point this function have to test.

Definition at line 307 of file TracksOfPoints.cpp.

Referenced by OpencvSfM::StructureEstimator::removeOutliersTracks().

4.17.2.16 void OpencvSfM::TrackOfPoints::set3DPosition (cv::Vec3d *newPoint*) [inline]

Use this function to change the 3D coordinates corresponding to this track

Parameters

newPoint	new 3D coordinates

Definition at line 172 of file TracksOfPoints.h.

4.17.2.17 void OpencvSfM::TrackOfPoints::setColor (unsigned int *c*) [inline]

Use this function to change the color of this track

Parameters

c	new color (ARGB packed into a int)

Definition at line 194 of file TracksOfPoints.h.

Referenced by OpencvSfM::SequenceAnalyzer::read().

4.17.2.18 DMatch OpencvSfM::TrackOfPoints::toDMatch (const int *img1*, const int *img2*) const

use this function to create a DMatch value from this track

Parameters

img1	train match image
img2	query match image

Returns

DMatch value

Definition at line 115 of file TracksOfPoints.cpp.

Referenced by OpencvSfM::EuclideanEstimator::initialReconstruction().

4.17.2.19 double OpencvSfM::TrackOfPoints::triangulateLinear (std::vector< PointOfView > & *cameras*, const std::vector< cv::Ptr< PointsToTrack > > & *points_to_track*, cv::Vec3d & *points3D*, const std::vector< bool > & *masks* = std::vector<bool>( ))

Using cameras and 2D points, try to find the 3D coordinates

Parameters

cameras	cameras used to compute projection of 3D point.
points_to_-track	2D points used to compute projection
points3D	3D coordinates of this tracks
masks	used to knwo which point this function have to use.

Definition at line 181 of file TracksOfPoints.cpp.

Referenced by triangulateRobust().

4.17.2.20 double OpencvSfM::TrackOfPoints::triangulateRobust (std::vector< PointOfView > & *cameras*, const std::vector< cv::Ptr< PointsToTrack > > & *points_to_track*, cv::Vec3d & *points3D*, double *reproj_error* = 4, const std::vector< bool > & *masks* = std::vector<bool>( ))

Using cameras and 2D points, try to find the best 3D coordinates which minimize the reprojection error using a RANSAC estimation

Parameters

cameras	cameras used to compute projection of 3D point.
points_to_-track	2D points used to compute projection
points3D	3D coordinates of the best estimation
reproj_error	Threshold used to reject outliners
masks	used to knwo which point this function have to use.

Definition at line 238 of file TracksOfPoints.cpp.

Referenced by OpencvSfM::StructureEstimator::computeStructure(), and OpencvSfM-::StructureEstimator::removeOutliersTracks().

4.17.3 Member Data Documentation

4.17.3.1 std::vector<bool> OpencvSfM::TrackOfPoints::good_values [protected]

Sometimes a 2d point is not good... This vector help us to know which points are correct...

Definition at line 36 of file TracksOfPoints.h.

Referenced by addMatch(), containPoint(), removeOutliers(), and OpencvSfM::-SequenceAnalyzer::write().

4.17.3.2 int OpencvSfM::TrackOfPoints::track_consistance [protected]

if <0 the track is inconsistent if >0 represent the degree of consistence (higher is better)

Definition at line 41 of file TracksOfPoints.h.

Referenced by addMatch(), OpencvSfM::SequenceAnalyzer::read(), and OpencvSfM::-SequenceAnalyzer::write().

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

- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/TracksOfPoints.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/TracksOfPoints.cpp

4.18 OpencvSfM::Visualizer Class Reference

This class can be used to view the differents object involved in current structure from motion process.

`#include <Visualizer.h>`

Public Member Functions

- Visualizer (std::string name="3D Viewer")
- void addCamera (const PointOfView &camera, std::string name="camera", int viewport=0)
- void add3DPoints (const std::vector< cv::Vec3d > &points, std::string name="cloud", int viewport=0)
- void add3DPointsColored (const std::vector< cv::Vec3d > &points, const std::vector< unsigned int > &colors, std::string name="cloud", int viewport=0)
- void runInteract ()

Protected Attributes

- boost::shared_ptr < pcl::visualization::PCLVisualizer > viewer

  The PCL viewer.

4.18.1 Detailed Description

This class can be used to view the differents object involved in current structure from motion process.

You can add to visualization 3D points, cameras, pictures... This class use PCL as back end, but it's hidden!

Definition at line 20 of file Visualizer.h.

4.18.2 Constructor & Destructor Documentation

4.18.2.1 OpencvSfM::Visualizer::Visualizer (std::string *name* = "3D Viewer")

Use this constructor to create a new window

Parameters

name	The title of the new window

Definition at line 14 of file Visualizer.cpp.

4.18.3 Member Function Documentation

4.18.3.1 void OpencvSfM::Visualizer::add3DPoints (const std::vector< cv::Vec3d > & *points*, std::string *name* = "cloud", int *viewport* = 0)

Use this function to add a new point cloud to the visualizer

Parameters

points	list of 3d points
name	The name of the printed object
viewport	idx of the wanted viewport

Definition at line 103 of file Visualizer.cpp.

4.18.3.2 void OpencvSfM::Visualizer::add3DPointsColored (const std::vector< cv::Vec3d > & *points*, const std::vector< unsigned int > & *colors*, std::string *name* = "cloud", int *viewport* = 0)

Use this function to add a new point cloud with color to the visualizer

Parameters

points	list of 3d points
colors	list of colors (RGB packed)
name	The name of the printed object
viewport	idx of the wanted viewport

Definition at line 127 of file Visualizer.cpp.

Referenced by OpencvSfM::EuclideanEstimator::viewEstimation().

4.18.3.3 void OpencvSfM::Visualizer::addCamera (const PointOfView & *camera*, std::string *name* = "camera", int *viewport* = 0)

Use this function to add a new camera to the visualizer

Parameters

camera	info about the wanted camera
name	The name of the printed object
viewport	idx of the wanted viewport

Definition at line 22 of file Visualizer.cpp.

Referenced by OpencvSfM::EuclideanEstimator::viewEstimation().

4.18.3.4 void OpencvSfM::Visualizer::runInteract ( )

Once geometry is added, you can used this function to enable user interaction

Definition at line 165 of file Visualizer.cpp.

Referenced by OpencvSfM::EuclideanEstimator::viewEstimation().

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

- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/-Visualizer.h
- D:/Travail/These/Determination caracteristiques camera/GSoC/SfM/src/-Visualizer.cpp

Index