Algoritmo di tracking per una camera monoculare

Generated by Doxygen 1.8.6

Fri Apr 15 2016 17:21:53

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 Camera Class Reference

`#include <Camera_code.h>`

Classes

- struct Obj

Public Member Functions

- Camera ()
  : constructor
- ~Camera ()
  : destructor
- void ControllCamera ()
  : Function that handle the first iteration.
- void DetectWithSift ()
  : Function that calculates the keypoint for the second image, and match it with the first image
- void Triangulation ()
  : Function that triangulate the point

Public Attributes

- ros::NodeHandle nh
- cv::Mat Camera_Matrix
- cv::Mat Cam_par_distortion
- cv::Mat Camera2_S03
- KDL::Frame Move_robot
- cv::Mat scene
- int arrived_cam = 0
- ros::Subscriber movewebcamrobot
- ros::Subscriber ptam_kf3d
- image_transport::ImageTransport it_
- image_transport::Subscriber sub
- ros::Subscriber ptam_sub
- struct Camera::Obj BottonCHosen

- std::vector< cv::Point2f > KeypointIm2
- std::vector< cv::Point2f > KeyPointIm1Match
- KDL::Frame frame_so3_ptam
- std::string camera_topic_
- bool move_camera_end
- int FirstCalibration
- cv::Mat frame1_
- double scala
- KDL::Frame So3_prev_ptam
- bool SaveFirst
- bool sub_ptam_2
- float cam_fx
- float cam_d0
- float cam_d1
- float cam_d2
- float cam_d3
- float cam_d4
- float cam_fy
- float cam_cx
- float cam_cy
- pcl::PointCloud< pcl::PointXYZ > Ptamkf3d
- KDL::Frame frame_w_c
- int count_n_passi

  camere in word

Static Public Attributes

- static cv::Point pos_object
- static int press_buttom = 0
- static int first_Step = 1

Private Member Functions

- void ShapeDetect ()

  : Function that find the botton desired
- std::pair< int, bool > FindAMinDistanceButton (std::vector< cv::Point > &baricentro, cv::Point &point_)
- void ImageConverter (const sensor_msgs::Image::ConstPtr &msg)

  : Callback that convert sensor image with cv mat
- std::pair< std::vector < cv::Point >,std::vector < std::vector< cv::Point > > > FindContours (cv::Mat bw, cv::Mat camera)

  : Approximate contour with accuracy proportional to the contour perimeter
- void SOtreCamera (const geometry_msgs::PoseWithCovarianceStamped::ConstPtr msg)

  : Pose word into camera
- void RobotMove (const geometry_msgs::Pose msg)

  : Callback that save the real movement of robot
- void InfoKf3d (const sensor_msgs::PointCloud2::ConstPtr &msg)

  : Callback that convert PointCloud2 to poincloudXYX
- void ProjectPointAndFindPosBot3d (std::vector< cv::Point3d > vect3d)

  : Reproject the points 3d to 2d and find the 3d botton pose
- void FillCamMatrixPose (KDL::Frame frame)
- std::vector< cv::Point3d > ConvertPointFromWordToCam ()

  : Convert each point in camera frame
- void FindBottonPos3D (Eigen::Vector4f plane_param)

  : Find the 3d botton pose

Static Private Member Functions

- static void CallBackFunc (int event, int x, int y, int flags, void *userdata)
  : Function that save the 2d position of press botton

3.1.1 Detailed Description

test

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Camera::Camera ( )

: constructor

3.1.2.2 Camera::~Camera ( ) [inline]

: destructor

3.1.3 Member Function Documentation

3.1.3.1 void Camera::CallBackFunc (int *event,* int *x,* int *y,* int *flags,* void * *userdata*) [static], [private]

: Function that save the 2d position of press botton

3.1.3.2 void Camera::ControllCamera ( )

: Function that handle the first iteration.

3.1.3.3 std::vector< cv::Point3d > Camera::ConvertPointFromWordToCam ( ) [private]

: Convert each point in camera frame

Returns

: 3d point in camera frame

Scaling each point with the scale factor.

3.1.3.4 void Camera::DetectWithSift ( )

: Function that calculates the keypoint for the second image, and match it with the first image

3.1.3.5 void Camera::FillCamMatrixPose (KDL::Frame *frame*) [private]

3.1.3.6 std::pair< int, bool > Camera::FindAMinDistanceButton (std::vector< cv::Point > & *baricentro,* cv::Point & *point_*) [private]

3.1.3.7 void Camera::FindBottonPos3D (Eigen::Vector4f *plane_param*) [private]

: Find the 3d botton pose

Parameters

in		plane param

Project the 2d botton position into 3d plane

3.1.3.8 std::pair< std::vector< cv::Point >,std::vector< std::vector< cv::Point > > > Camera::FindContours (cv::Mat *bw,* cv::Mat *camera*) [private]

: Approximate contour with accuracy proportional to the contour perimeter

Returns

: Center of each shape and his contours

3.1.3.9 void Camera::ImageConverter (const sensor_msgs::Image::ConstPtr & *msg*) [private]

: Callback that convert sensor image with cv mat

3.1.3.10 void Camera::InfoKf3d (const sensor_msgs::PointCloud2::ConstPtr & *msg*) [private]

: Callback that convert PointCloud2 to poincloudXYX

This pose is relative to word frame.

3.1.3.11 void Camera::ProjectPointAndFindPosBot3d (std::vector< cv::Point3d > *vect3d*) [private]

: Reproject the points 3d to 2d and find the 3d botton pose

Parameters

	pose of each feature

To find the 3d botton pose, we fit a plane with lms method and project the 2d botton position into this plane

3.1.3.12 void Camera::RobotMove (const geometry_msgs::Pose *msg*) [private]

: Callback that save the real movement of robot

This pose is relative to frame early

3.1.3.13 void Camera::ShapeDetect ( ) [private]

: Function that find the botton desired

3.1.3.14 void Camera::SOtreCamera (const geometry_msgs::PoseWithCovarianceStamped::ConstPtr *msg*) [private]

: Pose word into camera

3.1.3.15 void Camera::Triangulation ( )

: Function that triangulate the point

3.1.4 Member Data Documentation

3.1.4.1 int Camera::arrived_cam = 0

3.1.4.2 struct Camera::Obj Camera::BottonCHosen

3.1.4.3 float Camera::cam_cx

3.1.4.4 float Camera::cam_cy

3.1.4.5 float Camera::cam_d0

3.1.4.6 float Camera::cam_d1

3.1.4.7 float Camera::cam_d2

3.1.4.8 float Camera::cam_d3

3.1.4.9 float Camera::cam_d4

3.1.4.10 float Camera::cam_fx

3.1.4.11 float Camera::cam_fy

3.1.4.12 cv::Mat Camera::Cam_par_distortion

3.1.4.13 cv::Mat Camera::Camera2_S03

3.1.4.14 cv::Mat Camera::Camera_Matrix

3.1.4.15 std::string Camera::camera_topic_

3.1.4.16 int Camera::count_n_passi

camere in word

3.1.4.17 int Camera::first_Step = 1 [static]

3.1.4.18 int Camera::FirstCalibration

3.1.4.19 cv::Mat Camera::frame1_

3.1.4.20 KDL::Frame Camera::frame_so3_ptam

3.1.4.21 KDL::Frame Camera::frame_w_c

3.1.4.22 image_transport::ImageTransport Camera::it_

3.1.4.23 std::vector<cv::Point2f> Camera::KeyPointIm1Match

3.1.4.24 std::vector<cv::Point2f> Camera::KeypointIm2

3.1.4.25 bool Camera::move_camera_end

3.1.4.26 KDL::Frame Camera::Move_robot

3.1.4.27 ros::Subscriber Camera::movewebcamrobot

3.1.4.28 ros::NodeHandle Camera::nh

3.1.4.29 cv::Point Camera::pos_object [static]

3.1.4.30 int Camera::press_buttom = 0 [static]

3.1.4.31 ros::Subscriber Camera::ptam_kf3d

3.1.4.32 ros::Subscriber Camera::ptam_sub

3.1.4.33 pcl::PointCloud<pcl::PointXYZ> Camera::Ptamkf3d

3.1.4.34 bool Camera::SaveFirst

3.1.4.35 double Camera::scala

3.1.4.36 cv::Mat Camera::scene

3.1.4.37 KDL::Frame Camera::So3_prev_ptam

3.1.4.38 image_transport::Subscriber Camera::sub

3.1.4.39 bool Camera::sub_ptam_2

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

- /home/daniela/code/src/eye_in_hand/include/Camera_code.h
- /home/daniela/code/src/eye_in_hand/include/geometry_function.hpp
- /home/daniela/code/src/eye_in_hand/include/plane_estimate.hpp
- /home/daniela/code/src/eye_in_hand/src/Camera_code.cpp

3.2 Camera::Obj Struct Reference

`#include <Camera_code.h>`

Public Attributes

- std::vector< cv::Point > Bot_C
- cv::Point Center_
- std::vector< cv::KeyPoint > keyp_
- cv::Mat descr_
- cv::Mat figure_
- cv::Point3d Pos3d_
- cv::Point Botton_2frame

3.2.1 Member Data Documentation

3.2.1.1 std::vector<cv::Point> Camera::Obj::Bot_C

3.2.1.2 cv::Point Camera::Obj::Botton_2frame

3.2.1.3 cv::Point Camera::Obj::Center_

3.2.1.4 cv::Mat Camera::Obj::descr_

3.2.1.5 cv::Mat Camera::Obj::figure_

3.2.1.6 std::vector<cv::KeyPoint> Camera::Obj::keyp_

3.2.1.7 cv::Point3d Camera::Obj::Pos3d_

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

- /home/daniela/code/src/eye_in_hand/include/Camera_code.h

3.3 Settings Class Reference

`#include <calibration.h>`

Public Types

- enum Pattern { NOT_EXISTING, CHESSBOARD, CIRCLES_GRID, ASYMMETRIC_CIRCLES_GRID }
- enum InputType { INVALID, CAMERA, VIDEO_FILE, IMAGE_LIST }

Public Member Functions

- Settings ()
- void write (FileStorage &fs) const
- void read (const FileNode &node)
- void interprate ()
- Mat nextImage ()

Static Public Member Functions

- static bool readStringList (const string &filename, vector< string > &l)

Public Attributes

- Size boardSize
- Pattern calibrationPattern
- float squareSize
- int nrFrames
- float aspectRatio
- int delay
- bool bwritePoints
- bool bwriteExtrinsics
- bool calibZeroTangentDist
- bool calibFixPrincipalPoint
- bool flipVertical
- string outputFileName
- bool showUndistorsed
- string input
- int cameraID

- vector< string > imageList
- int atImageList
- VideoCapture inputCapture
- InputType inputType
- bool goodInput
- int flag

Private Attributes

- string patternToUse

3.3.1 Member Enumeration Documentation

3.3.1.1 enum Settings::InputType

Enumerator

INVALID

CAMERA

VIDEO_FILE

IMAGE_LIST

3.3.1.2 enum Settings::Pattern

Enumerator

NOT_EXISTING

CHESSBOARD

CIRCLES_GRID

ASYMMETRIC_CIRCLES_GRID

3.3.2 Constructor & Destructor Documentation

3.3.2.1 Settings::Settings ( ) [inline]

3.3.3 Member Function Documentation

3.3.3.1 void Settings::interprate ( ) [inline]

3.3.3.2 Mat Settings::nextImage ( ) [inline]

3.3.3.3 void Settings::read (const FileNode & *node*) [inline]

3.3.3.4 static bool Settings::readStringList (const string & *filename,* vector< string > & *l*) [inline], [static]

3.3.3.5 void Settings::write (FileStorage & *fs*) const [inline]

3.3.4 Member Data Documentation

3.3.4.1 float Settings::aspectRatio

3.3.4.2 int Settings::atImageList

3.3.4.3 Size Settings::boardSize

3.3.4.4 bool Settings::bwriteExtrinsics

3.3.4.5 bool Settings::bwritePoints

3.3.4.6 bool Settings::calibFixPrincipalPoint

3.3.4.7 Pattern Settings::calibrationPattern

3.3.4.8 bool Settings::calibZeroTangentDist

3.3.4.9 int Settings::cameraID

3.3.4.10 int Settings::delay

3.3.4.11 int Settings::flag

3.3.4.12 bool Settings::flipVertical

3.3.4.13 bool Settings::goodInput

3.3.4.14 vector<string> Settings::imageList

3.3.4.15 string Settings::input

3.3.4.16 VideoCapture Settings::inputCapture

3.3.4.17 InputType Settings::inputType

3.3.4.18 int Settings::nrFrames

3.3.4.19 string Settings::outputFileName

3.3.4.20 string Settings::patternToUse [private]

3.3.4.21 bool Settings::showUndistorsed

3.3.4.22 float Settings::squareSize

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

- /home/daniela/code/src/eye_in_hand/include/calibration.h

Chapter 4

File Documentation

4.1 /home/daniela/code/src/eye_in_hand/include/calibration.h File Reference

```
#include <iostream>
#include <sstream>
#include <time.h>
#include <stdio.h>
#include <opencv2/core/core.hpp>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/calib3d/calib3d.hpp>
#include <opencv2/highgui/highgui.hpp>
```

Classes

- class Settings

Macros

- #define _CRT_SECURE_NO_WARNINGS

Enumerations

- enum { DETECTION = 0, CAPTURING = 1, CALIBRATED = 2 }

Functions

- bool runCalibrationAndSave (Settings &s, Size imageSize, Mat &cameraMatrix, Mat &distCoeffs, vector< vector< Point2f > > imagePoints)

4.1.1 Macro Definition Documentation

4.1.1.1 #define _CRT_SECURE_NO_WARNINGS

4.1.2 Enumeration Type Documentation

4.1.2.1 anonymous enum

Enumerator

DETECTION

CAPTURING

CALIBRATED

4.1.3 Function Documentation

4.1.3.1 bool runCalibrationAndSave (Settings & *s*, Size *imageSize*, Mat & *cameraMatrix*, Mat & *distCoeffs*, vector< vector< Point2f > > *imagePoints*)

4.2 /home/daniela/code/src/eye_in_hand/include/Camera_code.h File Reference

```
#include <utility>
#include <list>
#include <string>
#include <eigen3/Eigen/Dense>
#include <kdl/frames.hpp>
#include <kdl_parser/kdl_parser.hpp>
#include <kdl/kdl.hpp>
#include <kdl/frames_io.hpp>
#include <ptam_com/KeyFrame_msg.h>
#include <ros/ros.h>
#include <image_transport/image_transport.h>
#include <cv_bridge/cv_bridge.h>
#include <sensor_msgs/image_encodings.h>
#include <opencv2/imgproc/imgproc.hpp>
#include <geometry_msgs/Pose.h>
#include <opencv2/highgui/highgui.hpp>
#include <std_msgs/Bool.h>
#include <sensor_msgs/PointCloud2.h>
#include <stdio.h>
#include <iostream>
#include <fstream>
#include "opencv2/core/core.hpp"
#include "opencv2/features2d/features2d.hpp"
#include "opencv2/nonfree/features2d.hpp"
#include "opencv2/nonfree/nonfree.hpp"
#include <tf/transform_broadcaster.h>
#include <geometry_msgs/PoseWithCovarianceStamped.h>
#include "opencv2/calib3d/calib3d.hpp"
#include <tf_conversions/tf_kdl.h>
#include <pcl/io/pcd_io.h>
#include <pcl_conversions/pcl_conversions.h>
#include <std_msgs/Float32MultiArray.h>
#include <visualization_msgs/MarkerArray.h>
```

Classes

- class Camera
- struct Camera::Obj

Functions

- Eigen::Vector4f EstimatePlane (std::vector< cv::Point3d > Point_Near)
  : Plane estimation
- void setLabel (cv::Mat &im, const std::string label, std::vector< cv::Point > &contour)
- cv::Point FindACenter (std::vector< cv::Point > &geometry)
  : Find a centroid of shape
- std::pair< int, int > FindMaxValue (cv::Mat &matrix, cv::Point &point)
  : function that calculates the max value of width and height for the roi
- double Media (cv::Mat triangulatedPoints3D, double MaxLenght, int col)
- void FromCvPointToEigen (cv::Point3d point_, Eigen::VectorXd &vect)
  : Convert cv point into eigen vector
- void FromMatToEigen (cv::Mat Mat_, Eigen::MatrixXd &Eigen)
  : Convert cv mat into eigen matrix
- void FromEigenVectorToCvPOint (Eigen::VectorXd Eigen, cv::Point3d &mat)
  : Convert eigen vector to cv point
- double ScalaReturn (double ptam, double ptam_prev, double robot)
  : Calculates the scale factor

4.2.1 Function Documentation

4.2.1.1 Eigen::Vector4f EstimatePlane (std::vector< cv::Point3d > *Point_Near*)

: Plane estimation

Parameters

in		3d position of each point near the botton

Estimate the plane with lms method

4.2.1.2 cv::Point FindACenter (std::vector< cv::Point > & *geometry*)

: Find a centroid of shape

Parameters

in		shape contours

Returns

: center position

4.2.1.3 std::pair<int,int> FindMaxValue (cv::Mat & *matrix,* cv::Point & *point*)

: function that calculates the max value of width and height for the roi

Parameters

in		mat of scene
in		point of interest

Returns

: range to cut the image to make a roi

4.2.1.4 void FromCvPointToEigen (cv::Point3d *point_*, Eigen::VectorXd & *vect*)

: Convert cv point into eigen vector

Parameters

in		cv::Point3d
out		Eigen::Vectorxd homogeneus

4.2.1.5 void FromEigenVectorToCvPOint (Eigen::VectorXd *Eigen,* cv::Point3d & *mat*)

: Convert eigen vector to cv point

Parameters

in		Eigen::VectorXd
out		cv::Point3d

4.2.1.6 void FromMatToEigen (cv::Mat *Mat_,* Eigen::MatrixXd & *Eigen*)

: Convert cv mat into eigen matrix

Parameters

in		cv::Mat
out		Eigen::MatrixXd homogeneus

4.2.1.7 double Media (cv::Mat *triangulatedPoints3D,* double *MaxLenght,* int *col*)

4.2.1.8 double ScalaReturn (double *ptam,* double *ptam_prev,* double *robot*)

: Calculates the scale factor

Parameters

in		actual position
in		previusly position
in		robot movements

Returns

scale factor

4.2.1.9 void setLabel (cv::Mat & *im,* const std::string *label,* std::vector< cv::Point > & *contour*)

4.3 /home/daniela/code/src/eye_in_hand/include/converter_file.hpp File Reference

Functions

- void FromMatToEigen (cv::Mat Mat_, Eigen::MatrixXd &Eigen1)

  : Convert cv mat into eigen matrix
- void FromCvPointToEigen (cv::Point3d point_, Eigen::VectorXd &vect)

  : Convert cv point into eigen vector
- void FromEigenVectorToCvPOint (Eigen::VectorXd Eigen, cv::Point3d &mat)

  : Convert eigen vector to cv point

4.3.1 Function Documentation

4.3.1.1 void FromCvPointToEigen (cv::Point3d *point_*, Eigen::VectorXd & *vect*)

: Convert cv point into eigen vector

Parameters

in		cv::Point3d
out		Eigen::Vectorxd homogeneus

4.3.1.2 void FromEigenVectorToCvPOint (Eigen::VectorXd *Eigen,* cv::Point3d & *mat*)

: Convert eigen vector to cv point

Parameters

in		Eigen::VectorXd
out		cv::Point3d

4.3.1.3 void FromMatToEigen (cv::Mat *Mat_,* Eigen::MatrixXd & *Eigen*)

: Convert cv mat into eigen matrix

Parameters

in		cv::Mat
out		Eigen::MatrixXd homogeneus

4.4 /home/daniela/code/src/eye_in_hand/include/geometry_function.hpp File Reference

Functions

- std::pair< int, int > FindMaxValue (cv::Mat &matrix, cv::Point &point)
  : function that calculates the max value of width and height for the roi
- cv::Point FindACenter (std::vector< cv::Point > &geometry)
  : Find a centroid of shape
- void setLabel (cv::Mat &im, const std::string label, std::vector< cv::Point > &contour)

4.4.1 Function Documentation

4.4.1.1 cv::Point FindACenter (std::vector< cv::Point > & *geometry*)

: Find a centroid of shape

Parameters

in		shape contours

Returns

: center position

4.4.1.2 std::pair<int,int> FindMaxValue (cv::Mat & *matrix,* cv::Point & *point*)

: function that calculates the max value of width and height for the roi

Parameters

in		mat of scene
in		point of interest

Returns

: range to cut the image to make a roi

4.4.1.3 void setLabel (cv::Mat & *im,* const std::string *label,* std::vector< cv::Point > & *contour*)

4.5 /home/daniela/code/src/eye_in_hand/include/plane_estimate.hpp File Reference

```
#include <pcl/search/kdtree.h>
#include <pcl/point_cloud.h>
#include <pcl/io/pcd_io.h>
#include <pcl/point_types.h>
#include <pcl/features/normal_3d.h>
```

Functions

- Eigen::Vector4f EstimatePlane (std::vector< cv::Point3d > Point_Near)

  : Plane estimation

4.5.1 Function Documentation

4.5.1.1 Eigen::Vector4f EstimatePlane (std::vector< cv::Point3d > *Point_Near*)

: Plane estimation

Parameters

in		3d position of each point near the botton

Estimate the plane with lms method

4.6 /home/daniela/code/src/eye_in_hand/src/calibration.cpp File Reference

```
#include <calibration.h>
```

Functions

- int main (int argc, char *argv[])
- bool runCalibrationAndSave (Settings &s, Size imageSize, Mat &cameraMatrix, Mat &distCoeffs, vector< vector< Point2f > > imagePoints)

4.6.1 Function Documentation

4.6.1.1 int main (int *argc,* char * *argv[]*)

4.6.1.2 bool runCalibrationAndSave (Settings & *s,* Size *imageSize,* Mat & *cameraMatrix,* Mat & *distCoeffs,* vector< vector< Point2f > > *imagePoints*)

4.7 /home/daniela/code/src/eye_in_hand/src/Camera_code.cpp File Reference

```
#include <Camera_code.h>
#include <geometry_function.hpp>
#include <converter_file.hpp>
#include <plane_estimate.hpp>
```

Functions

- cv::RNG rng (12345)
- int main (int argc, char **argv)
- double ScalaReturn (double ptam, double ptam_prev, double robot)

  : Calculates the scale factor

4.7.1 Function Documentation

4.7.1.1 int main (int *argc,* char ** *argv*)

4.7.1.2 cv::RNG rng (12345)

4.7.1.3 double ScalaReturn (double *ptam,* double *ptam_prev,* double *robot*)

: Calculates the scale factor

Parameters

in		actual position
in		previusly position
in		robot movements

Returns

scale factor

4.8 /home/daniela/code/src/eye_in_hand/src/main_image_.cpp File Reference

```
#include <ros/ros.h>
#include <image_transport/image_transport.h>
#include <cv_bridge/cv_bridge.h>
#include <stdio.h>
#include <iostream>
#include <opencv2/imgproc/imgproc.hpp>
#include <opencv2/highgui/highgui.hpp>
```

Functions

- int main (int argc, char **argv)

4.8.1 Function Documentation

4.8.1.1 int main (int *argc,* char ** *argv*)

Index