PerceptIn Robotics Vision System

API Documentation V0.4.0

April 2017

Contents

1 Namespace Index **2**

1.1 Namespace List 2

2 Hierarchical Index **3**

2.1 Class Hierarchy 3

3 Class Index **4**

3.1 Class List 4

4 Namespace Documentation **5**

4.1 PIRVS Namespace Reference 5

4.1.1 Detailed Description 6

4.1.2 Enumeration Type Documentation 6

4.1.2.1 SlamConfig 6

4.1.3 Function Documentation 7

4.1.3.1 CreatePerceptInV1Device() 7

4.1.3.2 Draw2dFeatures() 7

4.1.3.3 DrawStereoFeatures() 7

4.1.3.4 InitFeatureState() 8

4.1.3.5 InitMap() 9

4.1.3.6 InitState() 9

4.1.3.7 LoadMap() 10

4.1.3.8 RunFeature() 10

4.1.3.9 RunSlam() 11

4.1.3.10 RunTracking() 11

4.1.3.11 SaveMap() 12

5 Class Documentation **13**

5.1 PIRVS::Data Struct Reference . . . 13

5.1.1 Detailed Description . . . 13

5.2 PIRVS::DataLoader Class Reference . . . 13

5.2.1 Detailed Description . . . 14

5.2.2 Constructor & Destructor Documentation . . . 14

5.2.2.1 DataLoader() . . . 14

5.2.3 Member Function Documentation . . . 14

5.2.3.1 LoadData() . . . 14

5.3 PIRVS::FeatureState Class Reference . . . 15

5.3.1 Detailed Description . . . 15

5.3.2 Member Function Documentation . . . 15

5.3.2.1 Get2dFeatures() . . . 15

5.3.2.2 GetStereoFeatures() . . . 16

5.4 PIRVS::ImuData Struct Reference . . . 16

5.4.1 Detailed Description . . . 17

5.5 PIRVS::Map Class Reference . . . 17

5.5.1 Detailed Description . . . 17

5.5.2 Member Function Documentation . . . 17

5.5.2.1 GetPoints() . . . 17

5.6 PIRVS::PerceptInDevice Class Reference . . . 18

5.6.1 Detailed Description . . . 18

5.6.2 Member Function Documentation . . . 18

5.6.2.1 GetData() . . . 19

5.6.2.2 GetExposure() . . . 19

5.6.2.3 GUI() . . . 20

5.6.2.4 SetExposure() . . . 20

5.6.2.5 StartDevice() . . . 20

5.6.2.6 StartRecording() . . . 21

5.6.2.7 StopDevice() . . . 21

5.6.2.8 StopRecording() 22
5.7 PIRVS::SlamState Class Reference 22
5.7.1 Detailed Description 22
5.7.2 Member Function Documentation 23
5.7.2.1 GetPose() 23
5.8 PIRVS::StereoData Struct Reference 23
5.8.1 Detailed Description 24
5.9 PIRVS::StereoFeature Struct Reference 24
5.9.1 Detailed Description 24
5.10 PIRVS::TrajectoryDrawer Class Reference 24
5.10.1 Detailed Description 25
5.10.2 Constructor & Destructor Documentation 25
5.10.2.1 TrajectoryDrawer() 25
5.10.3 Member Function Documentation 25
5.10.3.1 Draw() 25

Index **27**

Chapter 1

Namespace Index

1.1 Namespace List

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

PIRVS 5

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

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

PIRVS::Data 13
PIRVS::ImuData 16
PIRVS::StereoData 23
PIRVS::DataLoader 13
PIRVS::FeatureState 15
PIRVS::Map 17
PIRVS::PerceptInDevice 18
PIRVS::SlamState 22
PIRVS::StereoFeature 24
PIRVS::TrajectoryDrawer 24

Chapter 3

Class Index

3.1 Class List

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

PIRVS::Data 13
PIRVS::DataLoader 13
PIRVS::FeatureState 15
PIRVS::ImuData 16
PIRVS::Map 17
PIRVS::PerceptInDevice 18
PIRVS::SlamState 22
PIRVS::StereoData 23
PIRVS::StereoFeature 24
PIRVS::TrajectoryDrawer 24

Chapter 4

Namespace Documentation

4.1 PIRVS Namespace Reference

Classes

- struct Data
- class DataLoader
- class FeatureState
- struct ImuData
- class Map
- class PerceptInDevice
- class SlamState
- struct StereoData
- struct StereoFeature
- class TrajectoryDrawer

Typedefs

- typedef size_t **Timestamp**

Enumerations

- enum SlamConfig { ONLINE_SLAM_CONFIG, OFFLINE_SLAM_CONFIG }

Functions

- bool InitFeatureState (const std::string &file_calib, std::shared_ptr< FeatureState > *state_ptr)

  Create an initial FeatureState.
- void RunFeature (std::shared_ptr< const StereoData > stereo_data, std::shared_ptr< FeatureState > state, bool with_3d)

  Process a StereoData and then update a FeatureState accordingly.
- bool Draw2dFeatures (std::shared_ptr< const StereoData > stereo_data, std::shared_ptr< const FeatureState > state, cv::Mat *img)

  Draw the detected 2d features for both left and right image.

- bool DrawStereoFeatures (std::shared_ptr< const StereoData > stereo_data, std::shared_ptr< const FeatureState > state, cv::Mat *img)

  Draw the stereo features on both left and right image.
- bool InitMap (const std::string &file_calib, const std::string &file_voc, std::shared_ptr< Map > *map_ptr)

  Create an initial Map to be used in RunSlam().
- bool LoadMap (const std::string &file_map, const std::string &file_calib, std::shared_ptr< Map > *map_ptr)

  Load a pre-built Map from disk to be used in RunTracking().
- bool SaveMap (const std::string &file_map, std::shared_ptr< const Map > map)

  Save a Map to disk.
- bool InitState (const std::string &file_calib, const SlamConfig config, std::shared_ptr< SlamState > *state_ptr)

  Create an initial SlamState to be used in RunSlam() and RunTracking().
- bool RunSlam (std::shared_ptr< const Data > data, std::shared_ptr< Map > map, std::shared_ptr< SlamState > state)

  Process a newly observed Data to update the Map and the SlamState.
- void RunTracking (std::shared_ptr< const Data > data, std::shared_ptr< const Map > map, std::shared_ptr< SlamState > state)

  Process a newly observed Data to the SlamState given a known Map.
- bool CreatePerceptInV1Device (std::shared_ptr< PerceptInDevice > *device_ptr)

  Create an interface for V1 PerceptInDevice.

4.1.1 Detailed Description

Copyright 2017 PerceptIn

This End-User License Agreement (EULA) is a legal agreement between you (the purchaser) and PerceptIn regarding your use of PerceptIn Robotics Vision System (PIRVS), including PIRVS SDK and associated documentation (the "Software").

IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS EULA, DO NOT INSTALL, USE OR COPY THE SOFTWARE.

4.1.2 Enumeration Type Documentation

4.1.2.1 SlamConfig

`enum PIRVS::SlamConfig`

Define different configurations for SLAM and tracking.

Enumerator

ONLINE_SLAM_CONFIG	Prefer speed over accuracy. Designed for online, real-time applications.
OFFLINE_SLAM_CONFIG	Prefer accuracy over speed. Designed for building an high-quality map from a recording.

4.1.3 Function Documentation

4.1.3.1 CreatePerceptInV1Device()

```
bool PIRVS::CreatePerceptInV1Device (
            std::shared_ptr< PerceptInDevice > * device_ptr )
```

Create an interface for V1 PerceptInDevice.

Parameters

out	*device_ptr*	Pointer to the shared_ptr of the newly created device interface.

Returns

True if the device is created.

4.1.3.2 Draw2dFeatures()

```
bool PIRVS::Draw2dFeatures (
            std::shared_ptr< const StereoData > stereo_data,
            std::shared_ptr< const FeatureState > state,
            cv::Mat * img )
```

Draw the detected 2d features for both left and right image.

This function produces a stereo image with the detected 2d features in blue hollow circles. The 2d location of these features can be queried from FeatureState::Get2dFeatures().

Parameters

	stereo_data	shared_ptr to the StereoData where the features are detected from.
	state	shared_ptr to the FeatureState updated by the `stereo_data`.
out	*img*	The output stereo image with the features overlay on top of it.

Returns

True if the `img` is created. False otherwise.

4.1.3.3 DrawStereoFeatures()

```
bool PIRVS::DrawStereoFeatures (
            std::shared_ptr< const StereoData > stereo_data,
```

```
            std::shared_ptr< const FeatureState > state,
            cv::Mat * img )
```

Draw the stereo features on both left and right image.

This function produces a stereo image with the stereo features overlay on top of it, color coded according to the depth (z value) of the 3d points. Blue represents a point with depth equal or less than 0.08 meter, and red represents depth with 4.0 meter and beyond. Depth between 0.08 and 4.0 are colored accordingly (i.e. green represents depth equal to 2.04 meter).

The stereo features can be queried from FeatureState::GetStereoFeatures().

Parameters

	stereo_data	shared_ptr to the StereoData where the features are detected from.
	state	shared_ptr to the FeatureState updated by the `stereo_data`.
`out`	*img*	The output stereo image with the features overlay on top of it.

Returns

True if the `img` is created. False otherwise.

4.1.3.4 InitFeatureState()

```
bool PIRVS::InitFeatureState (
            const std::string & file_calib,
            std::shared_ptr< FeatureState > * state_ptr )
```

Create an initial FeatureState.

Parameters

	file_calib	Path to the calibration (.json) file.
`out`	*state_ptr*	Pointer to the shared_ptr to the newly created FeatureState.

Returns

True if the state is created (i.e. the shared_ptr is not a nullptr). False otherwise. If false, the shared_ptr is nullptr.

Note

Common reasons for returning false includes, 1) the calibration file does not exist; 2) the calibration file is corrupted; 3) `state_ptr` is NULL.

4.1.3.5 InitMap()

```
bool PIRVS::InitMap (
            const std::string & file_calib,
            const std::string & file_voc,
            std::shared_ptr< Map > * map_ptr )
```

Create an initial Map to be used in RunSlam().

Parameters

	file_calib	Path to the calibration (.json) file.
	file_voc	Path to the vocabulary (.json) file.
out	*map_ptr*	Pointer to the shared_ptr of the newly created Map.

Returns

True if the Map is created. False if failed to create the Map, most likely because the `map_ptr` is NULL.

4.1.3.6 InitState()

```
bool PIRVS::InitState (
            const std::string & file_calib,
            const SlamConfig config,
            std::shared_ptr< SlamState > * state_ptr )
```

Create an initial SlamState to be used in RunSlam() and RunTracking().

Parameters

	file_calib	Path to the calibration (.json) file.
	config	The configuration for SLAM or tracking.
out	*state_ptr*	Pointer to the shared_ptr of the newly created SlamState.

Returns

True if the SlamState is created. False otherwise.

Note

Common reasons for returning false includes, 1) the calibration file does not exist or is corrupted; 2) `state_ptr` is NULL.

4.1.3.7 LoadMap()

```
bool PIRVS::LoadMap (
            const std::string & file_map,
            const std::string & file_calib,
            std::shared_ptr< Map > * map_ptr )
```

Load a pre-built Map from disk to be used in RunTracking().

Parameters

	file_map	Path to the (.json) file where the Map is stored.
	file_calib	Path to the calibration (.json) file.
out	*map_ptr*	Pointer to the shared_ptr of the newly created Map.

Returns

True if the Map is loaded. False if failed.

Note

Common reasons for returning false includes,p 1) the map file does not exist or is corrupted; 2) the calibration file does not exist or is corrupted; 3) `map_ptr` is NULL.

4.1.3.8 RunFeature()

```
void PIRVS::RunFeature (
            std::shared_ptr< const StereoData > stereo_data,
            std::shared_ptr< FeatureState > state,
            bool with_3d )
```

Process a StereoData and then update a FeatureState accordingly.

This function is the main function of the feature processing system, which detects 2d features from left and right images, matches features between the two images, and finally, triangulate the matched features to create 3d points.

Use FeatureState::Get2dFeatures() to query the detected 2d features and FeatureState::GetStereoFeatures() to query matched features with their triangulated 3d points.

Parameters

stereo_data	shared_ptr to the StereoData to process.
state	shared_ptr to the FeatureState to be updated.
with_3d	A flag to specify whether to compute the 3d points or not. If true the function will match the 2d features between the left and right images, and then, triangulate to get a 3d point. If false, this function will only detect 2d features.

4.1.3.9 RunSlam()

```
bool PIRVS::RunSlam (
            std::shared_ptr< const Data > data,
            std::shared_ptr< Map > map,
            std::shared_ptr< SlamState > state )
```

Process a newly observed Data to update the Map and the SlamState.

This function is the main function for SLAM (simultaneous localization/tracking and mapping).

Create the initial SlamState using InitState(), and create the initial Map using InitMap() to start SLAM.

After each RunSlam(), use SlamState::GetPose() to query the pose of the device, and use Map::GetPoints() to query the sparse 3d mapped points.

Parameters

	data	shared_ptr to the Data to process.
in,out	*map*	shared_ptr to the Map to be used to localize/track the device while being updated.
in,out	*state*	shared_ptr to the SlamState to update the pose of the device from.

Returns

True if the Map and SlamState are successfully update and the device is still "on track". False if any of the shared_ptr is nullptr, or if the SLAM algorithm failed to keep the device on track.

4.1.3.10 RunTracking()

```
void PIRVS::RunTracking (
            std::shared_ptr< const Data > data,
            std::shared_ptr< const Map > map,
            std::shared_ptr< SlamState > state )
```

Process a newly observed Data to the SlamState given a known Map.

This function is the main function for tracking the device within a known map.

Create the initial SlamState using InitState(), and load a pre-built Map using LoadMap().

After each RunTracking(), use SlamState::GetPose() to determine whether the device is still "on track" or not and to query the current pose of the device (if on track).

Parameters

	data	shared_ptr to the Data to process.
	map	shared_ptr to the Map to be used to localize/track the device while being updated.
in,out	*state*	shared_ptr to the SlamState to update the pose of the device from.

4.1.3.11 SaveMap()

```
bool PIRVS::SaveMap (
            const std::string & file_map,
            std::shared_ptr< const Map > map )
```

Save a Map to disk.

Parameters

file_map	Path to the (.json) file to write the Map to.
map	Pointer to the shared_ptr of the Map to store.

Returns

True if the Map is saved to disk. False otherwise.

Note

Common reasons for returning false includes, 1) `map` is nullptr; 2) `file_map` is empty.

Warning

This function will overwrite the file if `file_map` already exists.

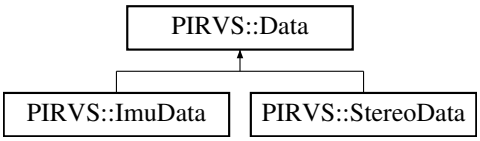
Chapter 5

Class Documentation

5.1 PIRVS::Data Struct Reference

`#include <pirvs.h>`

Inheritance diagram for PIRVS::Data:



Public Attributes

- Timestamp timestamp

  System timestamp at which the data is captured by the device.

5.1.1 Detailed Description

A container storing readings from a particular sensor in a PerceptIn device at a particular timestamp.

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

- include/pirvs.h

5.2 PIRVS::DataLoader Class Reference

`#include <pirvs.h>`

Public Member Functions

- DataLoader (const std::string &dir_data)

  Constructor.
- bool LoadData (std::shared_ptr< const Data > *data) const

  Load the next Data in the recorded sequence.

5.2.1 Detailed Description

An interface to load a recorded sequence from the disk.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 DataLoader()

```
PIRVS::DataLoader::DataLoader (
            const std::string & dir_data )
```

Constructor.

Parameters

dir_data	Directory where the data of the sequence are stored in disk.

5.2.3 Member Function Documentation

5.2.3.1 LoadData()

```
bool PIRVS::DataLoader::LoadData (
            std::shared_ptr< const Data > * data ) const
```

Load the next Data in the recorded sequence.

Parameters

out	*data*	Pointer to the shared_ptr to the loaded Data.

Returns

True if a Data is loaded. False (most likely) if the DataLoader reaches the end of the sequence, or if there are errors loading the next frame.

Note

Common errors includes: 1) `data` is NULL; 2) missing one of the images from the stereo camera; 3) the DataLoader wasn't created properly due to corrupted sequence.

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

- include/pirvs.h

5.3 PIRVS::FeatureState Class Reference

```
#include <pirvs.h>
```

Public Member Functions

- virtual bool Get2dFeatures (std::vector< cv::Point2d > *features_l, std::vector< cv::Point2d > *features_r) const =0

  Get the 2d feaatures detected from both sensors in of the stereo camera.
- virtual bool GetStereoFeatures (std::vector< StereoFeature > *features) const =0

  Get the stereo features detected from the current StereoData.

5.3.1 Detailed Description

State of the feature processing system.

Use InitFeatureState() to create a FeatureState. Note, InitFeatureState() is the only way to create a FeatureState.

Example:

```
std::shared_ptr<FeatureState> state;
InitFeatureState("calibration.json", &state);
```

5.3.2 Member Function Documentation

5.3.2.1 Get2dFeatures()

```
virtual bool PIRVS::FeatureState::Get2dFeatures (
            std::vector< cv::Point2d > * features_l,
            std::vector< cv::Point2d > * features_r ) const  [pure virtual]
```

Get the 2d feaatures detected from both sensors in of the stereo camera.

The stereo features are available from the FeatureState after it is updated via RunFeature().

Parameters

out	*features_l*	Pointer to the vector of 2d features detected by the left sensor of the stereo.
out	*features_r*	Pointer to the vector of 2d features detected by the right sensor of the stereo.

Returns

True if the features are available. False if the features are not available or the pointer `features` is NULL.

5.3.2.2 GetStereoFeatures()

```
virtual bool PIRVS::FeatureState::GetStereoFeatures (
            std::vector< StereoFeature > * features ) const  [pure virtual]
```

Get the stereo features detected from the current StereoData.

The stereo features are available from the FeatureState if it is updated via RunFeature() with 3d turned on.

Parameters

out	*features*	Pointer to the vector of stereo features.

Returns

True if the features are available. False if the features are not available or the pointer `features` is NULL.

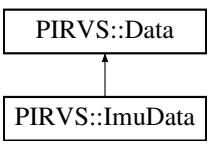
The documentation for this class was generated from the following file:

- include/pirvs.h

5.4 PIRVS::ImuData Struct Reference

```
#include <pirvs.h>
```

Inheritance diagram for PIRVS::ImuData:



Public Attributes

- cv::Vec3d accel

  Acceleration reading from the accelerometer. Unit: meter / sec^ 2.
- cv::Vec3d ang_v

  Angular velocity reading from the gyroscope. Unit: radian / sec.

5.4.1 Detailed Description

A container storing readings from the 6-dof IMU from a PerceptIn device.

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

- include/pirvs.h

5.5 PIRVS::Map Class Reference

```
#include <pirvs.h>
```

Public Member Functions

- virtual bool GetPoints (std::vector< cv::Point3d > *points) const =0

  Get the sparse 3d points in the map.

5.5.1 Detailed Description

3D map describing the geometric structure of the observed environment.

Use InitMap() to create an empty map, and then, call RunSlam() to incrementally build the Map while localizing the device within that Map. Use SaveMap() to store a Map to the disk. Use LoadMap() to load a pre-built Map from disk, and then call RunTracking() to localize the device within that Map.

The coordinate of the Map is defined so that the z-axis is pointing towards the gravity direction.

5.5.2 Member Function Documentation

5.5.2.1 GetPoints()

```
virtual bool PIRVS::Map::GetPoints (
            std::vector< cv::Point3d > * points ) const  [pure virtual]
```

Get the sparse 3d points in the map.

Parameters

out	*points*	Pointer to the vector of points.

Returns

False if `points` is NULL.

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

- include/pirvs.h

5.6 PIRVS::PerceptInDevice Class Reference

`#include <pirvs.h>`

Public Member Functions

- virtual bool StartDevice ()=0

  Start streaming from the device.
- virtual bool StopDevice ()=0

  Stop streaming from the device.
- virtual bool StartRecording (const std::string &dir="/tmp/")=0

  Start recording data from the device.
- virtual bool StopRecording (std::string *dir=nullptr, size_t *num_imu=nullptr, size_t *num_stereo=nullptr)=0

  Stop recording data from the device.
- virtual bool SetExposure (const uint32_t value)=0

  Set exposure of the stereo camera.
- virtual bool GetExposure (uint32_t *value) const =0

  Read the exposure value of the stereo camera from the device.
- virtual bool GetData (std::shared_ptr< const Data > *data_ptr)=0

  Get the latest Data from the device.
- virtual void GUI (const std::string &dir="/tmp/")=0

  Start the GUI to view live stream of the stereo camera. The GUI can also be used to record sequences and to play around with the exposure values (with a trackbar on the top).

5.6.1 Detailed Description

An interface to stream data from a PerceptIn device.

Use CreatePerceptInV1Device() to create an interface for V1 PerceptInDevice.

5.6.2 Member Function Documentation

5.6.2.1 GetData()

```
virtual bool PIRVS::PerceptInDevice::GetData (
            std::shared_ptr< const Data > * data_ptr )  [pure virtual]
```

Get the latest Data from the device.

This function is intended to be called in a while loop.

Example:

```
std::shared_ptr<PerceptInDevice> device;
// Create device here.
device->StartDevice();
while (condition) {
  std::shared_ptr<const Data> data;
  if (!device->GetData(&data)) {
    continue;
  }
  // Process data here.
}
```

Note if the data processing is too slow, this function will drop frames.

Parameters

out	*data_ptr*	Pointer to the shared_ptr of the newly obtained Data.

Returns

True if data is available. False otherwise.

Warning

Can not be called if GUI() is on.

5.6.2.2 GetExposure()

```
virtual bool PIRVS::PerceptInDevice::GetExposure (
            uint32_t * value ) const  [pure virtual]
```

Read the exposure value of the stereo camera from the device.

Parameters

out	*value*	The exposure value obtained from the device.

Returns

True if the `value` is available. False if failed to query the exposure value from device.

Warning

Can not be called if GUI() is on.

5.6.2.3 GUI()

```
virtual void PIRVS::PerceptInDevice::GUI (
            const std::string & dir = "/tmp/" )  [pure virtual]
```

Start the GUI to view live stream of the stereo camera. The GUI can also be used to record sequences and to play around with the exposure values (with a trackbar on the top).

Once the GUI launched, press space-bar to start recording, and then press space-bar again to stop recording. Each recording sequence is stored in a folder under `dir` with the name being the timestamp of the first data in the sequence.

Parameters

dir	Path to the root folder where the recorded sequence will be stored.

5.6.2.4 SetExposure()

```
virtual bool PIRVS::PerceptInDevice::SetExposure (
            const uint32_t value )  [pure virtual]
```

Set exposure of the stereo camera.

Parameters

value	The exposure value to set to.

Returns

True if the exposure value is set to `value`.

Warning

Can not be called if GUI() is on.

5.6.2.5 StartDevice()

```
virtual bool PIRVS::PerceptInDevice::StartDevice ( )  [pure virtual]
```

Start streaming from the device.

Returns

True if device starts streaming. False otherwise.

Warning

Can not be called if GUI() is on.

5.6.2.6 StartRecording()

```
virtual bool PIRVS::PerceptInDevice::StartRecording (
            const std::string & dir = "/tmp/" )  [pure virtual]
```

Start recording data from the device.

The caller provides a root folder to store the recorded data, and StartRecording() will create a folder under the root folder with the name being the timestamp of the first data in the sequence.

Parameters

dir	Path to the root folder where the recorded sequence will be stored.

Returns

True if recording starts. False otherwise.

Note

Common reasons for returning false includes, 1) device hasn't been started yet; 2) device is already recording data.

Warning

Can not be called if GUI() is on.

5.6.2.7 StopDevice()

```
virtual bool PIRVS::PerceptInDevice::StopDevice ( )  [pure virtual]
```

Stop streaming from the device.

Returns

True if device stops streaming. False otherwise.

Warning

Can not be called if GUI() is on.

5.6.2.8 StopRecording()

```
virtual bool PIRVS::PerceptInDevice::StopRecording (
            std::string * dir = nullptr,
            size_t * num_imu = nullptr,
            size_t * num_stereo = nullptr )  [pure virtual]
```

Stop recording data from the device.

Parameters

out	*dir*	Path to the folder where the recorded data sequence is stored.
out	*num_imu*	Number of IMU data recorded
out	*num_stereo*	Number of stereo data recorded.

Returns

True if the device stopped recording. False otherwise.

Note

Common reasons for returning false includes, 1) device hasn't been started yet; 2) device is not recording data;

Warning

Can not be called if GUI() is on.

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

- include/pirvs.h

5.7 PIRVS::SlamState Class Reference

```
#include <pirvs.h>
```

Public Member Functions

- virtual bool GetPose (cv::Affine3d *pose) const =0

  Get the current pose of the SLAM system.

5.7.1 Detailed Description

The state of the SLAM system at specific timestamp.

Use InitState() to create a SlamState.

5.7.2 Member Function Documentation

5.7.2.1 GetPose()

```
virtual bool PIRVS::SlamState::GetPose (
            cv::Affine3d * pose ) const  [pure virtual]
```

Get the current pose of the SLAM system.

The pose is represented by a 4-by-4 transformation that brings a 3d point from the Map coordinate to the coordinate of the device. The pose is available if the device is "on track" (i.e. the device is properly localized within the Map).

Example:

```
// Point in the Map coordinate.
cv::Point2d p_map;
// Same point in the device coordinate.
cv::Point2d p_device = pose * p_map;
```

Parameters

out	*pose*	Pointer to the pose.

Returns

True if pose is available. False otherwise.

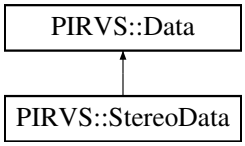
The documentation for this class was generated from the following file:

- include/pirvs.h

5.8 PIRVS::StereoData Struct Reference

`#include <pirvs.h>`

Inheritance diagram for PIRVS::StereoData:



Public Attributes

- cv::Mat img_l

  The image captured from the left sensor of the stereo camera.
- cv::Mat img_r

  The image associated from the left sensor of the stereo camera.

5.8.1 Detailed Description

A container storing readings from the stereo camera from a PerceptIn device.

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

- include/pirvs.h

5.9 PIRVS::StereoFeature Struct Reference

`#include <pirvs.h>`

Public Attributes

- cv::Point2d pt_l

  2d image location of the point in the left sensor of the stereo camera.
- cv::Point2d pt_r

  2d image location of the point in the right sensor of the stereo camera.
- cv::Point3d pt_3d

  3D point in the left camera's coordinate.

5.9.1 Detailed Description

A 3d point seen by both sensors in the stereo camera in the PerceptIn device.

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

- include/pirvs.h

5.10 PIRVS::TrajectoryDrawer Class Reference

`#include <pirvs.h>`

Public Member Functions

- TrajectoryDrawer (const size_t img_size=500)

  Constructor.
- bool Draw (std::shared_ptr< const SlamState > state, cv::Mat *img)

  Update the drawer according to the latest SlamState, and produce an image with the latest pose.

5.10.1 Detailed Description

An interface to visualize trajectory of the tracking pose of the device.

This interface visualizes the 6-dof pose from the top down view. Thus, the x-axis and the y-axis of the global map are visualized but not the z-axis. Note, the z-axis is pointing towards the gravity direction.

The center of the view is the origin of the global coordinate. The x-axis is drawn in blue and the y-axis in red. The length of the axis equals to 1 meter in the physical world. Note, the interface will zoom out as the device moves further away from origin.

The current pose of the device is visualized by a circle centered at the (x, y) location of the device and a line pointing out from the circle indicating the heading direction of the device. The trajectory of the past 3 seconds is visualized in a green.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 TrajectoryDrawer()

```
PIRVS::TrajectoryDrawer::TrajectoryDrawer (
            const size_t img_size = 500 )
```

Constructor.

Parameters

img_size	Size of the squared image to visualize the trajectory on.

5.10.3 Member Function Documentation

5.10.3.1 Draw()

```
bool PIRVS::TrajectoryDrawer::Draw (
            std::shared_ptr< const SlamState > state,
            cv::Mat * img )
```

Update the drawer according to the latest SlamState, and produce an image with the latest pose.

Parameters

	state	shared_ptr to the latest SlamState.
out	*img*	Pointer to the image with the visualization. The size of the image is specified in the constructor.

Returns

True if the image is created. False if the `state` is nullptr or `img` is NULL.

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

- include/pirvs.h

Index

CreatePerceptInV1Device
    PIRVS, 7

DataLoader
    PIRVS::DataLoader, 14
Draw
    PIRVS::TrajectoryDrawer, 25
Draw2dFeatures
    PIRVS, 7
DrawStereoFeatures
    PIRVS, 7

GUI
    PIRVS::PerceptInDevice, 20
Get2dFeatures
    PIRVS::FeatureState, 15
GetData
    PIRVS::PerceptInDevice, 18
GetExposure
    PIRVS::PerceptInDevice, 19
GetPoints
    PIRVS::Map, 17
GetPose
    PIRVS::SlamState, 23
GetStereoFeatures
    PIRVS::FeatureState, 16

InitFeatureState
    PIRVS, 8
InitMap
    PIRVS, 8
InitState
    PIRVS, 9

LoadData
    PIRVS::DataLoader, 14
LoadMap
    PIRVS, 9

PIRVS::Data, 13
PIRVS::DataLoader, 13
    DataLoader, 14
    LoadData, 14
PIRVS::FeatureState, 15
    Get2dFeatures, 15
    GetStereoFeatures, 16
PIRVS::ImuData, 16
PIRVS::Map, 17
    GetPoints, 17
PIRVS::PerceptInDevice, 18
    GUI, 20
    GetData, 18
    GetExposure, 19
    SetExposure, 20
    StartDevice, 20
    StartRecording, 21
    StopDevice, 21
    StopRecording, 21
PIRVS::SlamState, 22
    GetPose, 23
PIRVS::StereoData, 23
PIRVS::StereoFeature, 24
PIRVS::TrajectoryDrawer, 24
    Draw, 25
    TrajectoryDrawer, 25
PIRVS, 5
    CreatePerceptInV1Device, 7
    Draw2dFeatures, 7
    DrawStereoFeatures, 7
    InitFeatureState, 8
    InitMap, 8
    InitState, 9
    LoadMap, 9
    RunFeature, 10
    RunSlam, 10
    RunTracking, 11
    SaveMap, 12
    SlamConfig, 6

RunFeature
    PIRVS, 10
RunSlam
    PIRVS, 10
RunTracking
    PIRVS, 11

SaveMap
    PIRVS, 12
SetExposure
    PIRVS::PerceptInDevice, 20
SlamConfig
    PIRVS, 6
StartDevice
    PIRVS::PerceptInDevice, 20
StartRecording
    PIRVS::PerceptInDevice, 21
StopDevice
    PIRVS::PerceptInDevice, 21
StopRecording
    PIRVS::PerceptInDevice, 21

TrajectoryDrawer
PIRVS::TrajectoryDrawer, 25