Physical line follower robot, learning with Closed-Loop Deep Learning (CLDL)

Generated by Doxygen 1.8.19

Chapter 1

serialib class

The class serialib offers simple access to the serial port devices for windows and linux. It can be used for any serial device (Built-in serial port, USB to RS232 converter, arduino board or any hardware using or emulating a serial port) The class can be used under Windows and Linux. The class allows basic operations like :

- opening and closing connection
- reading data (characters, array of bytes or strings)
- writing data (characters, array of bytes or strings)
- non-blocking functions (based on timeout).

Author

Philippe Lucidarme (University of Angers) serialib@googlegroups.com

Date

1th may 2011 (Last update: 25th september 2012)

Version

1.2

Chapter 2

LineFollowerRobot

This repository provides a platform for online predictive learning in the context of closed-loop robotic systems. The physical robot is built on a SumoBot chassis with a mounted Raspberry Pi that serves as a computation engine for the learning algorithm. The camera provides a vision of the road ahead for prediction. The steering command from RPi is passed to an Arduino that generates the PWM signal for the robot's servo motors. The Light sensors from the Robot provide instructive feedback to the learner in the form of a closed-loop error signal.

2.0.1 Building LineFollowerRobot

LineFollowerRobot has the following dependencies that must be installed:

- `boost`
- `opencv`

In order to build:

- enter the LineFollowerRobot directory – `cd lineFollowerRobot`
- run cmake – `cmake .`
- run the build system – `make`

2.0.2 The robot

Chapter 3

Class Index

3.1 Class List

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

Chapter 4

File Index

4.1 File List

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

Chapter 5

Class Documentation

5.1 Bandpass Class Reference

`#include <bandpass.h>`

Public Member Functions

- Bandpass ()
- double filter (double v)
- void calcPolesZeros (double f, double r)
- void setParameters (double frequency, double Qfactor)
- void impulse (char *name)
- void calcNorm (double f)
- void transfer (char *name)
- double getOutput ()
- void reset ()

5.1.1 Detailed Description

Creates memory traces at specified length. It's a 2nd order IIR filter.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 Bandpass()

`Bandpass::Bandpass ( )`

Constructor

5.1.3 Member Function Documentation

5.1.3.1 calcNorm()

```
void Bandpass::calcNorm (
            double f )
```

Normalises the output with f

5.1.3.2 calcPolesZeros()

```
void Bandpass::calcPolesZeros (
            double f,
            double r )
```

Calculates the coefficients The frequency is the normalized frequency in the range [0..0.5].

5.1.3.3 filter()

```
double Bandpass::filter (
            double v )
```

Filter

5.1.3.4 getOutput()

```
double Bandpass::getOutput ( )  [inline]
```

Gets the output of the filter. Same as the return value of the function "filter()".

5.1.3.5 impulse()

```
void Bandpass::impulse (
            char * name )
```

Generates an acsii file with the impulse response of the filter.

5.1.3.6 reset()

```
void Bandpass::reset ( )
```

Sets the output to zero again

5.1.3.7 setParameters()

```
void Bandpass::setParameters (
            double frequency,
            double Qfactor )
```

sets the filter parameters

5.1.3.8 transfer()

```
void Bandpass::transfer (
            char * name )
```

Generates an ASCII file with the transfer function

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

- bandpass.h
- bandpass.cpp

5.2 cvui::cvui_block_t Struct Reference

Public Attributes

- cv::Mat **where**
- cv::Rect **rect**
- cv::Rect **fill**
- cv::Point **anchor**
- int **padding**
- int **type**

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

- cvui.h

5.3 cvui::cvui_context_t Struct Reference

Public Attributes

- cv::String **windowName**
- cvui_mouse_t **mouse**

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

- cvui.h

5.4 cvui::cvui_label_t Struct Reference

Public Attributes

- bool **hasShortcut**
- char **shortcut**
- std::string **textBeforeShortcut**
- std::string **textAfterShortcut**

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

- cvui.h

5.5 cvui::cvui_mouse_btn_t Struct Reference

Public Attributes

- bool **justReleased**
- bool **justPressed**
- bool **pressed**

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

- cvui.h

5.6 cvui::cvui_mouse_t Struct Reference

Public Attributes

- cvui_mouse_btn_t **buttons** [3]
- cvui_mouse_btn_t **anyButton**
- cv::Point **position**

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

- cvui.h

5.7 Extern Class Reference

```
#include <external.h>
```

Public Member Functions

- Extern ()
- ~Extern ()
- int onStepCompleted (Mat &statFrame, double deltaSensorData, vector< double > &predictorDeltas)
- double calcError (Mat &statFrame, vector< uint8_t > &sensorCHAR)
- void calcPredictors (Mat &frame, vector< double > &predictorDeltaMeans)
- int getNpredictors ()

5.7.1 Detailed Description

Main class for robot communication

5.7.2 Constructor & Destructor Documentation

5.7.2.1 Extern()

```
Extern::Extern ( )
```

Constructor

5.7.2.2 ~Extern()

```
Extern::~Extern ( )
```

Destructor

5.7.3 Member Function Documentation

5.7.3.1 calcError()

```
double Extern::calcError (
            Mat & statFrame,
            vector< uint8_t > & sensorCHAR )
```

This function calculates the closed-loop error from the raw data that are received from the Arduino. It plots the data on the Stat Frame, it also calculates the integral error and monitors the 'success condition'.

Parameters

statFrame	The frame where the data is plotted
sensorCHAR	An array of Characters: the raw data from the ground sensors

Returns

Returns the closed-loop error

5.7.3.2 calcPredictors()

```
void Extern::calcPredictors (
            Mat & frame,
            vector< double > & predictorDeltaMeans )
```

This function calculates the predictor signals.

Parameters

frame	The camera view
predictorDeltaMeans	A pointer to an array where the predictor signals are stored

5.7.3.3 getNpredictors()

```
int Extern::getNpredictors ( )
```

It reports on the number of predictors (pixel clusters) used

Returns

Returns the number of predictors

5.7.3.4 onStepCompleted()

```
int Extern::onStepCompleted (
            Mat & statFrame,
            double deltaSensorData,
            vector< double > & predictorDeltas )
```

This is called at every time-step, it calls the neural network internally

Parameters

statFrame	The frame where the data is plotted
deltaSensorData	The error signal from the sensors
predictorDeltas	The predictor signals from the camera

Returns

returns the differential speed to be sent to the motors

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

- external.h
- external.cpp

5.8 LowPassFilter Class Reference

Public Member Functions

- **LowPassFilter** (float iCutOffFrequency)
- **LowPassFilter** (float iCutOffFrequency, float iDeltaTime)
- float **update** (float input)
- float **update** (float input, float deltaTime)
- float **getOutput** ()
- float **getCutOffFrequency** ()
- void **setCutOffFrequency** (float input)
- void **setDeltaTime** (float input)

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

- LowPassFilter.hpp
- LowPassFilter.cpp

5.9 serialib Class Reference

This class can manage a serial port. The class allows basic operations (opening the connection, reading, writing data and closing the connection).

```
#include <serialib.h>
```

Public Member Functions

- serialib ()

  Constructor of the class serialib.
- ~serialib ()

  Destructor of the class serialib. It close the connection.
- char Open (const char *Device, const unsigned int Bauds)

  Open the serial port.
- void Close ()

  Close the connection with the current device.
- char WriteChar (char)

  Write a char on the current serial port.
- char ReadChar (char *pByte, const unsigned int TimeOut_ms=0)

Wait for a byte from the serial device and return the data read.

- char WriteString (const char ∗String)

  Write a string on the current serial port.
- int ReadString (char ∗String, char FinalChar, unsigned int MaxNbBytes, const unsigned int TimeOut_ms=0)

  Read a string from the serial device (with timeout)
- char Write (const void ∗Buffer, const unsigned int NbBytes)

  Write an array of data on the current serial port.
- int Read (void ∗Buffer, unsigned int MaxNbBytes, const unsigned int TimeOut_ms=0)

  Read an array of bytes from the serial device (with timeout)
- void FlushReceiver ()

  Empty receiver buffer (UNIX only)
- int Peek ()

  Return the number of bytes in the received buffer (UNIX only)

5.9.1 Detailed Description

This class can manage a serial port. The class allows basic operations (opening the connection, reading, writing data and closing the connection).

5.9.2 Member Function Documentation

5.9.2.1 Open()

```
char serialib::Open (
             const char * Device,
             const unsigned int Bauds )
```

Open the serial port.

Parameters

Device	: Port name (COM1, COM2, ... for Windows) or (/dev/ttyS0, /dev/ttyACM0, /dev/ttyUSB0 ... for linux)

Parameters

Bauds	: Baud rate of the serial port. \n Supported baud rate for Windows : - 110 - 300 - 600 - 1200 - 2400 - 4800 - 9600 - 14400 - 19200 - 38400 - 56000 - 57600 - 115200 - 128000 - 256000 \n Supported baud rate for Linux :\n - 110 - 300 - 600 - 1200 - 2400 - 4800 - 9600 - 19200 - 38400 - 57600 - 115200

Returns

1 success

-1 device not found

-2 error while opening the device

-3 error while getting port parameters

-4 Speed (Bauds) not recognized

-5 error while writing port parameters

-6 error while writing timeout parameters

5.9.2.2 Peek()

```
int serialib::Peek ( )
```

Return the number of bytes in the received buffer (UNIX only)

Returns

The number of bytes in the received buffer

5.9.2.3 Read()

```
int serialib::Read (
            void * Buffer,
            unsigned int MaxNbBytes,
            const unsigned int TimeOut_ms = 0 )
```

Read an array of bytes from the serial device (with timeout)

Parameters

Buffer	: array of bytes read from the serial device
MaxNbBytes	: maximum allowed number of bytes read
TimeOut_ms	: delay of timeout before giving up the reading

Returns

1 success, return the number of bytes read

0 Timeout reached

-1 error while setting the Timeout

-2 error while reading the byte

5.9.2.4 ReadChar()

```
char serialib::ReadChar (
            char * pByte,
            const unsigned int TimeOut_ms = 0 )
```

Wait for a byte from the serial device and return the data read.

Parameters

pByte	: data read on the serial device
TimeOut_ms	: delay of timeout before giving up the reading If set to zero, timeout is disable (Optional)

Returns

1 success

0 Timeout reached

-1 error while setting the Timeout

-2 error while reading the byte

5.9.2.5 ReadString()

```
int serialib::ReadString (
              char * String,
              char FinalChar,
              unsigned int MaxNbBytes,
              const unsigned int TimeOut_ms = 0 )
```

Read a string from the serial device (with timeout)

Parameters

String	: string read on the serial device
FinalChar	: final char of the string
MaxNbBytes	: maximum allowed number of bytes read
TimeOut_ms	: delay of timeout before giving up the reading (optional)

Returns

>0 success, return the number of bytes read

0 timeout is reached

-1 error while setting the Timeout

-2 error while reading the byte

-3 MaxNbBytes is reached

5.9.2.6 Write()

```
char serialib::Write (
              const void * Buffer,
              const unsigned int NbBytes )
```

Write an array of data on the current serial port.

Parameters

Buffer	: array of bytes to send on the port
NbBytes	: number of byte to send

Returns

1 success

-1 error while writting data

5.9.2.7 WriteChar()

```
char serialib::WriteChar (
            char Byte )
```

Write a char on the current serial port.

Parameters

Byte	: char to send on the port (must be terminated by '\0')

Returns

1 success

-1 error while writting data

5.9.2.8 WriteString()

```
char serialib::WriteString (
            const char * String )
```

Write a string on the current serial port.

Parameters

String	: string to send on the port (must be terminated by '\0')

Returns

1 success

-1 error while writting data

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

- serialib.h
- serialib.cpp

5.10 TimeOut Class Reference

This class can manage a timer which is used as a timeout.

```
#include <serialib.h>
```

Public Member Functions

- TimeOut ()

  Constructor of the class TimeOut.
- void InitTimer ()

  Initialise the timer. It writes the current time of the day in the structure PreviousTime.
- unsigned long int ElapsedTime_ms ()

  Returns the time elapsed since initialization. It write the current time of the day in the structure CurrentTime. Then it returns the difference between CurrentTime and PreviousTime.

5.10.1 Detailed Description

This class can manage a timer which is used as a timeout.

5.10.2 Member Function Documentation

5.10.2.1 ElapsedTime_ms()

```
unsigned long int TimeOut::ElapsedTime_ms ( )
```

Returns the time elapsed since initialization. It write the current time of the day in the structure CurrentTime. Then it returns the difference between CurrentTime and PreviousTime.

Returns

The number of microseconds elapsed since the functions InitTimer was called.

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

- serialib.h
- serialib.cpp

5.11 cvui::internal::TrackbarParams Struct Reference

Public Attributes

- long double **min**
- long double **max**
- long double **step**
- int **segments**
- unsigned int **options**
- std::string **labelFormat**

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

- cvui.h

Chapter 6

File Documentation

6.1 serialib.cpp File Reference

Class to manage the serial port.

```
#include "serialib.h"
#include <cstdint>
```

6.1.1 Detailed Description

Class to manage the serial port.

Author

Philippe Lucidarme (University of Angers) serialib@googlegroups.com

Version

1.2

Date

28 avril 2011

6.2 serialib.h File Reference

Serial library to communicate throught serial port, or any device emulating a serial port.

Classes

- class serialib

  This class can manage a serial port. The class allows basic operations (opening the connection, reading, writing data and closing the connection).
- class TimeOut

  This class can manage a timer which is used as a timeout.

6.2.1 Detailed Description

Serial library to communicate throught serial port, or any device emulating a serial port.

Author

Philippe Lucidarme (University of Angers) serialib@googlegroups.com

Version

1.2

Date

28 avril 2011 This Serial library is used to communicate through serial port.

Chapter 7

Example Documentation

7.1 Example1.cpp