plotOpenCv C++ library

v1.0.2

Table of contents

Overview

plotOpenCv is a C++ library developed to facilitate the visualization of 2-dimensional line charts. This library is built upon the OpenCV, providing users with a convenient and efficient tool for visualizing data through line charts. With plotOpenCv, users can effortlessly create multiple line charts within a single window and tune various chart parameters, such as line width, color, and more. It uses C++17 standard. The library is licensed under the Apache 2.0 license.

Versions

Table 1 - Library versions.

Version	Release date	What's new
1.0.0	08.09.2023	First version.
1.0.1	18.09.2023	- Update used container for plots.

Version	Release date	What's new
1.0.2	16.04.2024	- Antialiased line drawing implemented. - Window size issue fixed. - Documentation updated.

Library files

The library is supplied only by source code. The user is given a set of files in the form of a CMake project (repository). The repository structure is shown below:

```
CMakeLists.txt -------------- Main CMake file of the library.
src ------------------------- Library source code folder.
    CMakeLists.txt ---------- CMake file of the library.
    plotOpenCv.h ------------ Main library header file.
    plotOpenCvVersion.h ----- Header file with library version.
    plotOpenCvVersion.h.in -- File for CMake to generate version header.
    plotOpenCv.cpp ---------- C++ implementation file.
test ------------------------ Folder for test application.
    CMakeLists.txt ---------- CMake file of test application.
    main.cpp ---------------- Source code of test application.
```

Plot class description

Class declaration

Plot class declared in **plotOpenCv.h** file. Class declaration:

```
class Plot
{
public:

    /// Get string of current library version.
    static std::string getVersion();

    /// Class constructor.
    Plot(std::string name, int width = 1280, int height = 720,
         cv::Scalar backgroundColor = cv::Scalar(255, 255, 255),
         cv::Scalar scaleLineColor = cv::Scalar(0, 128, 128));

    /// Class destructor.
    ~Plot();

    /// Render plots on window.
    template <typename T>
    void addPlot(std::vector<T>& points, int id, int start = 0, int end = 0,
                 cv::Scalar color = cv::Scalar(255, 255, 255), int thickness = 1);
    /// Method to render plots on window.
    template <typename T>
    void addPlot(std::vector<std::vector<T>>& points, int id,
                 int start = 0, int end = 0,
                 cv::Scalar color = cv::Scalar(255, 255, 255), int thickness = 1);

    /// Method to clean window.
    void clean();

    /// Method to show window.
    void show();
};
```

getVersion method

The **getVersion()** method returns string of current version of **plotOpenCv**. Method declaration:

```
static std::string getVersion();
```

Method can be used without **plotOpenCv** class instance:

```
std::cout << "plotOpenCv class version: " << plotOpenCv::getVersion() << std::endl;
```

Console output:

```
plotOpenCv class version: 1.0.2
```

addPlot (for 1D dataset) method

The **addPlot(...)** method serves the purpose of incorporating a new line chart into the existing window. It either introduces a new plot if the provided id is not yet present, or updates an existing plot associated with the given identifier. Method declaration:

```
void addPlot(std::vector<T> &points, int id, int start = 0, int end = 0,
            cv::Scalar color = cv::Scalar(255, 255, 255), int thickness = 1);
```

Parameter	Value
Points	One dimentional vector which includes vertical points.Vector format : {y1, y2, ... }
id	Identifier for chart on a window. Provides user to update a chart or add new one.
start	Start index of plot from vector when user wants to plot a specific range from a dataset. Should be 0 for whole dataset.
end	End index of plot from vector when user wants to plot a specific range from a dataset. Should be 0 for whole dataset.
color	Color of chart line.

Parameter	Value
thickness	Thickness of chart line.

addPlot (for 2D dataset) method

The **addPlot(...)** method serves the purpose of incorporating a new line chart into the existing window. It either introduces a new plot if the provided id is not yet present, or updates an existing plot associated with the given identifier. Method declaration:

```
void addPlot(std::vector<std::vector<T>> &points, int id, int start = 0, int end = 0,
             cv::Scalar color = cv::Scalar(255, 255, 255), int thickness = 1);
```

Parameter	Value
Points	Two dimentional vector which includes vertical and horizontal points. Vector format : {{x1,y1}, {x2,y2}, ... }
id	Identifier for chart on a window. Provides user to update a chart or add new one.
start	Start index of plot from vector when user wants to plot a specific range from a dataset. Should be 0 for whole dataset.
end	End index of plot from vector when user wants to plot a specific range from a dataset. Should be 0 for whole dataset.
color	Color of chart line.
tickness	Tickness of chart line.

Table 2 - Supported data types.

Supported data types
unsigned char
char
unsigned int
unsigned short
short int
int
float
double

show method

The **show()** method is responsible for displaying a window containing all the plotted line charts. Method declaration:

```
void show();
```

clean method

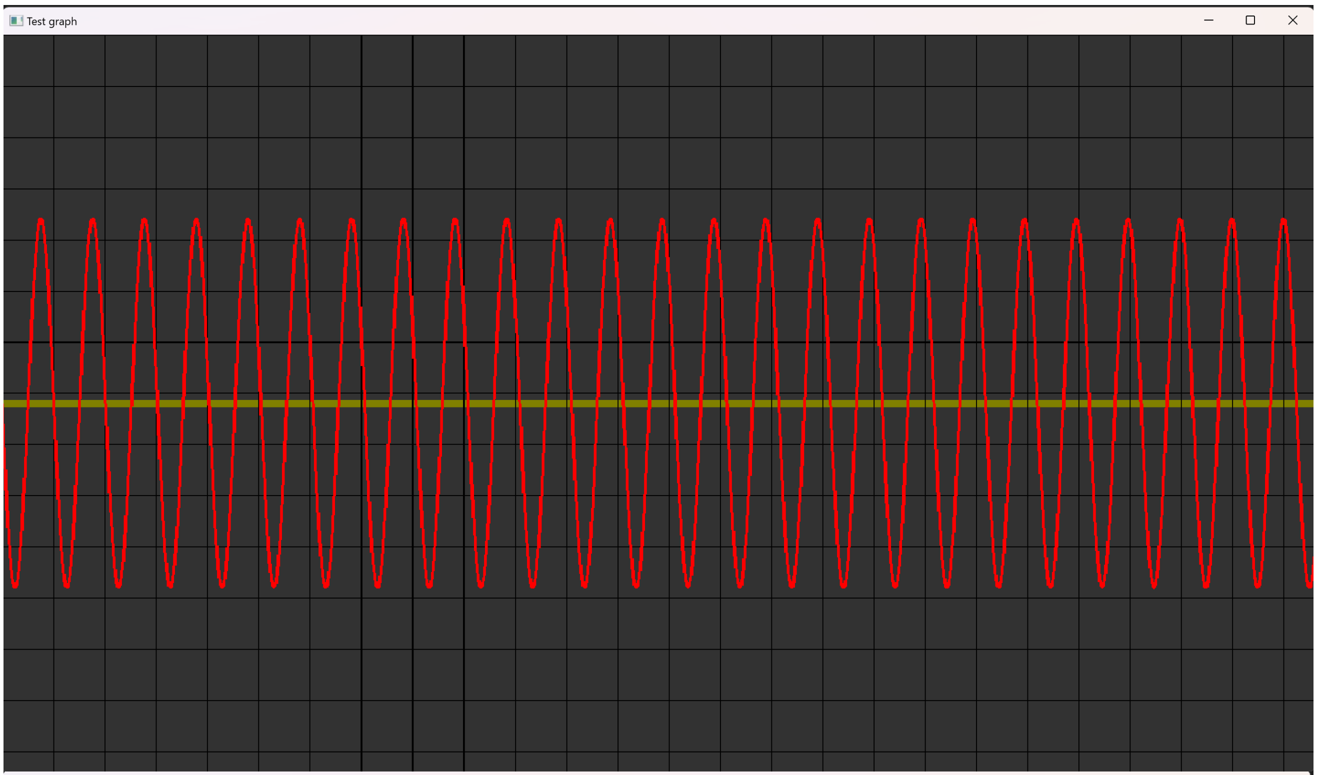
The **clean()** method is responsible for cleaning a window containing all the plotted line charts. Method declaration:

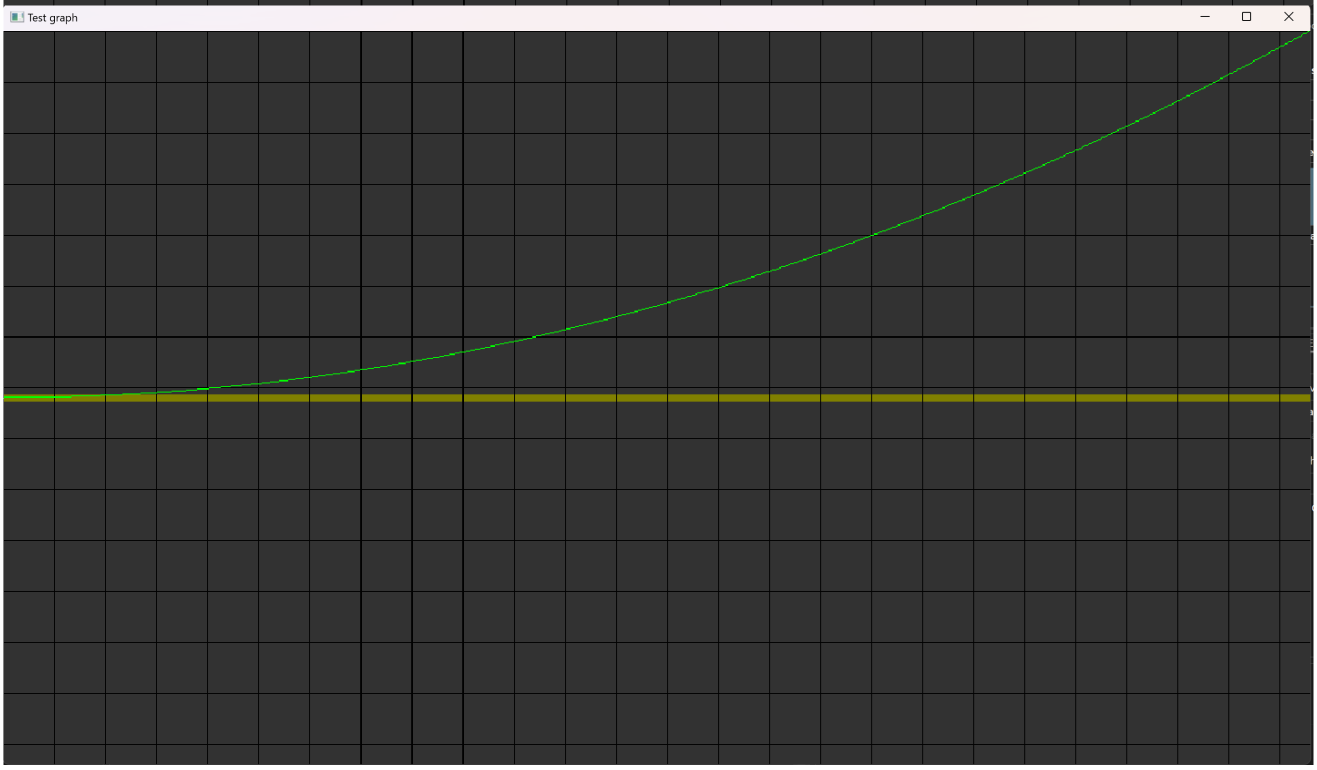
```
void clean();
```

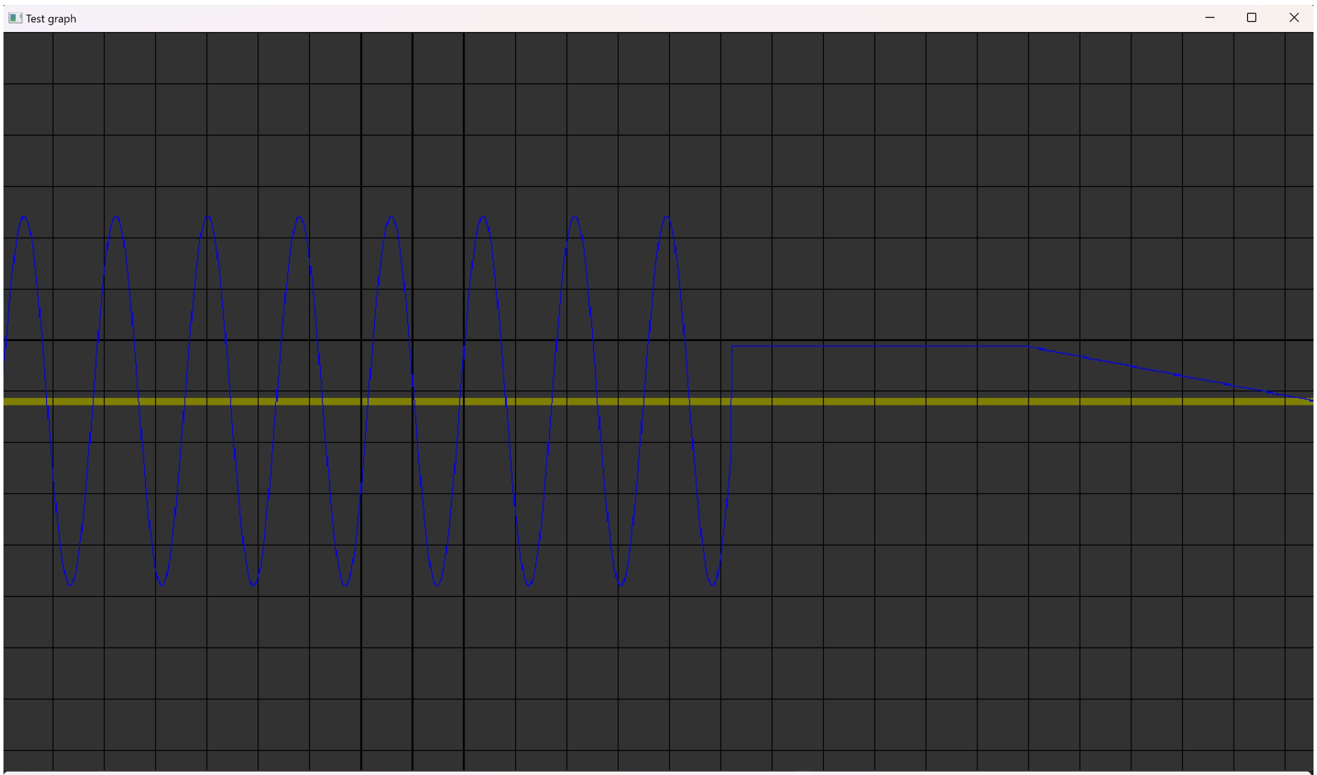
Example

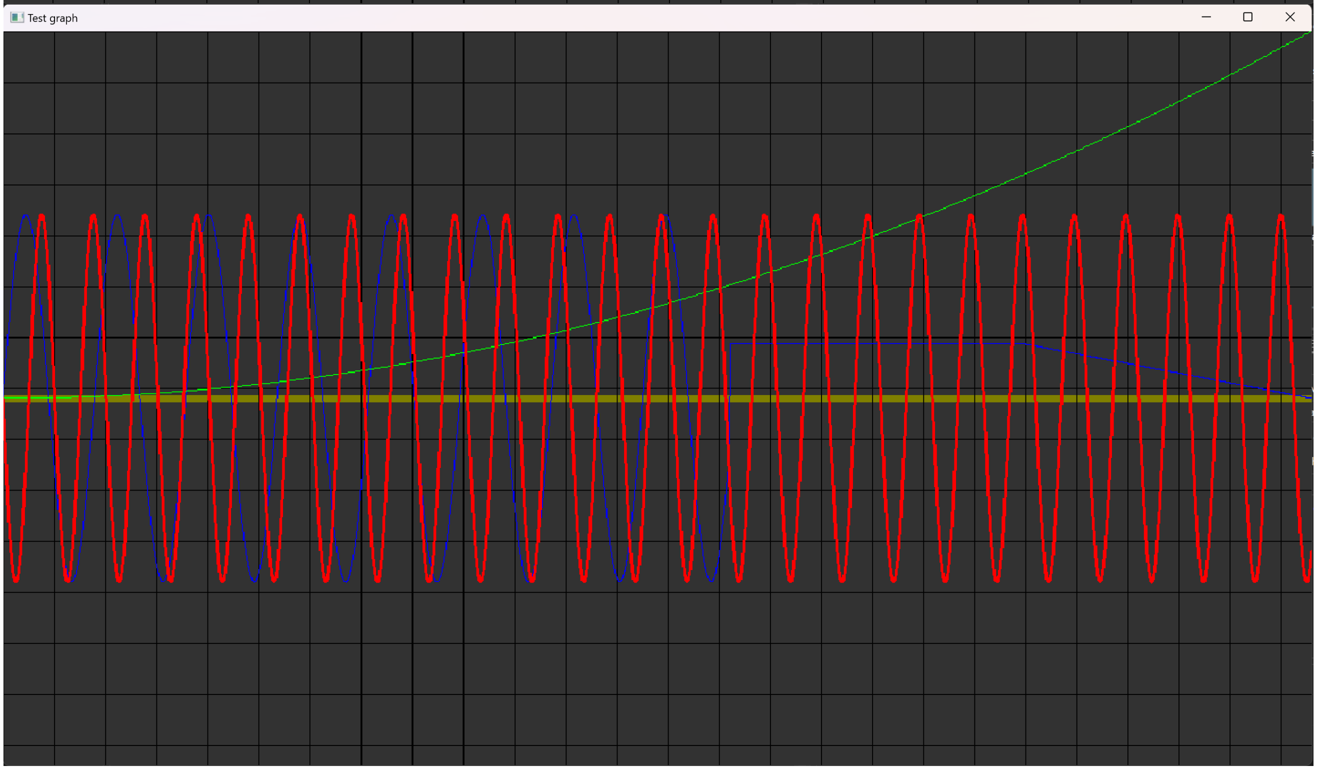
```
    plot graph("Test graph", 1280, 720,cv::Scalar(0, 128, 128) cv::Scalar(50, 50, 50));

    std::vector<float> linePoints(9000);
    std::vector<std::vector<float>> linePoints2(5000, std::vector<float>(2));

    graph.addPlot(linePoints,0, 0, 0, cv::Scalar(255,0,0), 5);
    graph.addPlot(linePoints2,1, 0, 0, cv::Scalar(0,255,0), 2);

    graph.show();
    cv::waitKey(0);
```

Example Charts


Test graph


Test graph





Build and connect to your project

Typical commands to build **plotOpenCv** library:

```
git clone https://github.com/ConstantRobotics-Ltd/plotOpenCv.git
cd plotOpenCv
mkdir build
cd build
cmake ..
make
```

If you want connect **plotOpenCv** library to your CMake project as source code you can make follow. For example, if your repository has structure:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
```

You can add repository **plotOpenCv** as submodule by commands:

```
cd <your respository folder>
git submodule add https://github.com/ConstantRobotics-Ltd/plotOpenCv.git
3rdparty/plotOpenCv
```

In you repository folder will be created folder **3rdparty/plotOpenCv** which contains files of **plotOpenCv** repository. New structure of your repository:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
3rdparty
    plotOpenCv
```

Create CMakeLists.txt file in **3rdparty** folder. CMakeLists.txt should contain:

```
cmake_minimum_required(VERSION 3.13)

################################################################################
## 3RD-PARTY
## dependencies for the project
################################################################################
project(3rdparty LANGUAGES CXX)

################################################################################
## SETTINGS
## basic 3rd-party settings before use
################################################################################
# To inherit the top-level architecture when the project is used as a submodule.
SET(PARENT ${PARENT}_YOUR_PROJECT_3RDPARTY)
# Disable self-overwriting of parameters inside included subdirectories.
SET(${PARENT}_SUBMODULE_CACHE_OVERWRITE OFF CACHE BOOL "" FORCE)

################################################################################
## CONFIGURATION
## 3rd-party submodules configuration
################################################################################
SET(${PARENT}_SUBMODULE_PLOT_OPENCV                     ON  CACHE BOOL "" FORCE)
if (${PARENT}_SUBMODULE_PLOT_OPENCV)
    SET(${PARENT}_PLOT_OPENCV                           ON  CACHE BOOL "" FORCE)
    SET(${PARENT}_PLOT_OPENCV_TEST                      OFF CACHE BOOL "" FORCE)
```

```
endif()

################################################################################
## INCLUDING SUBDIRECTORIES
## Adding subdirectories according to the 3rd-party configuration
################################################################################
if (${PARENT}_SUBMODULE_PLOT_OPENCV)
    add_subdirectory(plotOpenCv)
endif()
```

File **3rdparty/CMakeLists.txt** adds folder **plotOpenCv** to your project. Your repository new structure will be:

```
CMakeLists.txt
src
    CMakeList.txt
    yourLib.h
    yourLib.cpp
3rdparty
    CMakeLists.txt
    plotOpenCv
```

Next you need include folder 3rdparty in main **CMakeLists.txt** file of your repository. Add string at the end of your main **CMakeLists.txt**:

```
add_subdirectory(3rdparty)
```

Next you have to include plotOpenCv library in your **src/CMakeLists.txt** file:

```
target_link_libraries(${PROJECT_NAME} plotOpenCv)
```

Done!