Friedrich-Alexander-University Erlangen-Nuremberg

Chair of Multimedia Communications and Signal Processing

Prof. Dr.-Ing. André Kaup

Research Internship

Development of an Integrated Runtime Analyzing Toolkit for High Efficiency Video Coding (HEVC)

von Paulo André Camara Martinez

December 2017

Advisor: Nils Genser, M.Sc.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit ohne fremde Hilfe und ohne Benutzung anderer als der angegebenen Quellen angefertigt habe, und dass die Arbeit in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegen hat und von dieser als Teil einer Prüfungsleistung angenommen wurde. Alle Ausführungen, die wörtlich oder sinngemäß übernommen wurden, sind als solche gekennzeichnet.

_____ _____

Ort, Datum Unterschrift

Contents

Abstract

During development, debugging and evaluation of multimedia algorithms, displaying graphic data is especially helpful. And although popular in implementation of image and video standards, C++ lacks a good library for such capability. Most available solutions are complicated, have verbose usage and offer limited plotting options. Because of that, a new solution is needed to attend the need of displaying data generated by multimedia codecs, which has simple usage, minimal dependencies and a variety of features. In this research internship, a toolkit for analyzing data in the High Efficiency Video Codec (HEVC) is developed. It converts OpenCV data and configuration calls into MATLAB/Octave code and pipes it to the respective process, taking advantage of the rich range of plotting options both softwares provide. The library is also capable of dealing with multiple figures at once and easily generate data from simple external calls.

Chapter 1

Introduction

Current video coders consist of several steps and algorithms. Hence, a great amount of data is generated midway into execution. For purposes of debugging as well as research and development, this data is crucial for analyzing the codec performance.
Thus, the presence of a plotting solution would be a very helpful tool. However, there is no consolidated graphic library for C++. All current options are deficient in some sense. With that in mind, a new solution is required for fulfilling our purposes.
In this work, a toolkit for data visualization was developed. The C++ data is stored in OpenCV's *cv::Mat* type [1], then it is streamed to MATLAB [2] or octave [3], taking advantage of the variety of graphic functions provided by those softwares. It is also possible to easily generate data from MATLAB statements and control multiple figures.
This report is structured as follows: chapter 2 describes the initial considerations about choosing an appropriate path for the toolkit development; chapter 3 outlines the sturcture of the code nad overviews the offered features; chapter 5 details the process of choosing an appropriate license; finally, chapter 6 contains the conclusion.

Chapter 2

Alternatives

At the beginning, an evaluation of the current solutions available for C++ was required. Some known libraries include

- **PLplot** [4]
- **DISLIN** [5]
- **matplotlib-cpp** [6]
- **gnuplot-iostream** [7]

Regarding our requirements, we look for a solution that has simple usage - i.e., few lines of code for making a plot, a syntax that is similar to popular plotting software is a plus. Besides the need of support for traditional two-dimensional plots, three-dimensional surfaces, histograms and vector fields would also be desired. Furthermore, we aim for minimal dependencies, which should be reliable, widely distributed and easy to setup. Regarding the graphic data, it is going to be stored at *cv::Mat* type from OpenCV. Because of this, it should be simple to convert the data to a format the graphic solution understands.

Firstly, PLplot and DISLIN were considered. They are cross-platform software packages for scientific plots. Although having native integration, they are not widely dis-

tributed and their usage is too much verbose, requiring pixel control of most elements in the figure.

Turning to interface library solutions, matplotlib-ccp is an C++ interface for plotting with matplotlib/Python as an external engine. However, it only implements few functions from matplotlib. Another hindrance is that matplotlib itself has problems with engine configurations accross platforms, notably in windows.

Another option was gnuplot-iostream, which instead employs gnuplot, a ubiquitous command-line graphic utility that has been developed for decades. The library has straightforward usage and is composed by a single header file on C++. Its simplycity comes from the fact that it uses an iostream pipe to push plain gnuplot code to the external process. At first glance gnuplot-iostream seemed the best option, because it is uncomplicated and relies on a highly supported and stable project.

However, the library proved inefficient after a couple weeks of development. It has several problems in handling multiple threads and consequently multiple open figures. Thus, closing graphics upon termination as well as blocking the program execution in order to wait for the graphics was very difficult to achieve.

Because of that, we started exploring another solution. Instead of using gnuplot as an external engine for piping the data, we turned to MATLAB and Octave, which are popular options for displaying graphic data. Moreover, the syntax proximity between both programs means the same code can be used by both pipe processes.

Chapter 3

Structure

In this chapter, the library structure is explained. It fundamentally consists of four different classes:

- `PlotLib`
- `PlotLibConfig`
- `PlotLibData`
- `PlotLibFigure`

Firstly, `PlotLibConfig` stores the various configurations and styles for the plot as well as axis configurations. Besides methods for specifying common properties, such as title (`PlotLibConfig::set_title`) and axis labels (`PlotLibConfig::set_xlabel`), there is support for any graphic or axis configuration which can take the name-atribute format, by using functions `PlotLibConfig::add_kwargs` and `PlotLibConfig::add_ax_conf`, respectively. Furthermore, it can specify to which figure the plot should be generated. In the following we find a code snippet showing how an instance should be handled:

```
1 PlotLibConfig c;
2 c.set_title("Example");
3 c.set_xlabel("X-axis");
```

```
4 c.add_kwargs("MarkerFaceColor", "r");
5 c.add_ax_conf("XGrid", "on");
```

For handling the data to be used in the graphics, the PlotLibData class is employed. Each axis data is stored by using OpenCV's *cv::Mat* data type, which can be either one-dimensional vectors or two-diemensional matrices, dependending on the type of plot. Each axis has available methods to set, get and remove data. THe following code show the class can be employed:

```
1 cv::Mat x = cv::Mat(1, 10, CV_64FC1);
2 cv::Mat y = cv::Mat(1, 10, CV_64FC1, 1);
3
4 PlotLibData d;
5 d.set_x(x); d.set_y(y);
6 d.remove_x();
7 cv::Mat yy = d.get_y();
```

`PlotLib` is the toolkit's main class. All plotting functions and methods for handling the external engine process are declared here, as well as figures and hold options. Some methods are listed below:

```
1 PlotLib::sendCommand("r=magic(5);")
2
3 PlotLib::getData("1:.5:10")
4
5 PlotLib::readFromProcess()
6 PlotLib::close()
7
8 PlotLib::set_hold(c, "on")
9
10 PlotLib::plot(d, c)
11 PlotLib::scatter(d, c)
```

12 PlotLib::surf(d, c)

Regarding the external engine, it is possible to send commands, get data from declarations, read the stdout from the process and close the engine. Morever, the hold option for displaying multiple graphics at the same figure is present.
Finally, all graphic functions are implemented here. They take both an `PlotLibData` and `PlotLibConfig` object as arguments.
The following two-dimensional functions can be used from both MATLAB and octave:

- plot
- scatter
- stem
- loglog
- semilogx
- semilogy
- rect

Likewise, the library also offers support for three-dimensional functions, which are:

- plot3
- scatter3
- contour
- contour3
- surf
- mesh
- rectangle3 (draws two-dimensional rectangle at fixed z-axis value)

At last, the PlotLibFigure class is use for instantiating each figure, containing its ID and name. Each instance is stored into the *PlotLib::figures* dictionary, so all available figures can later be managed. If we want to plot into a already created figure, `PlotLibConfig::set_figure` should be used.

```
1 PlotLibConfig c;
2 PlotLib::new_figure("Some figure");
3 c.set_figure(1);
```

With this four classes, we have the full implementation of the library. In summary, the current toolkit capabilities include:

- Generate cv::Mat data from MATLAB declarations.
- Handle figures and axes.
- Access MATLAB/Octave graphic functions as well as run any command at the external engine.
- Customize plot style, such as title, axis labels, line specification and axis properties.

Chapter 4

Examples

In this chapter, we present a couple of plotting examples alongside its code. Figure 4.1 shows multiple two-dimensional functions being used at the same figure. The code for generating it is provided below:

```
1  cv::Mat x = PL::getData("1:10");
2  cv::Mat y = PL::getData("(1:10).^2");
3
4  PlotLibData d; d.set_x(x); d.set_y(y);
5
6  PlotLibConfig c;
7  PlotLib::new_figure();
8  c.set_figure(1);
9  c.set_title("Example...");
10 c.set_xlabel("X-axis");
11 c.set_ylabel("Y-axis");
12 c.add_ax_conf("xlim", "[0 20]");
13
14 PL::plot(d, c); PL::set_hold(c, "on");
15
```

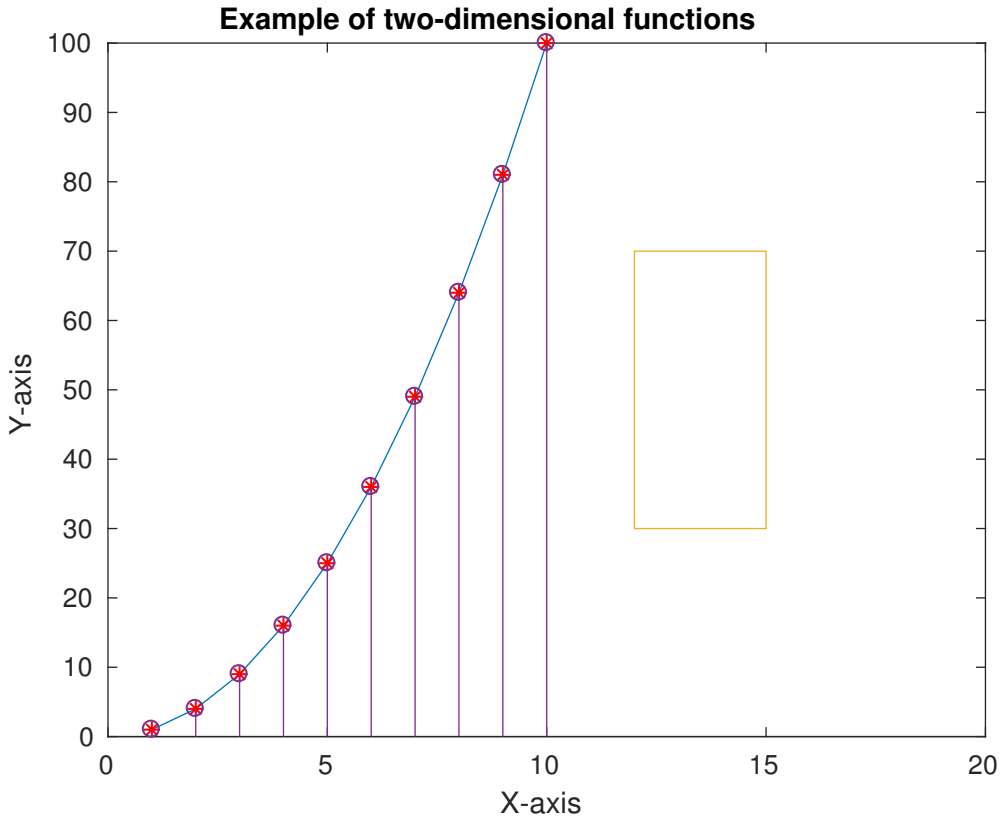


Figure 4.1: Example with functions *plot*, *scatter*, *stem* and *rect*

```
16 c_plot.set_linespec("*r");
17 PL::scatter(d, c);
18 c_plot.set_linespec("");
19
20 PL::rectangle(cv::Rect(12,30,3,40),c);
21 PL::stem(d, c); PL::set_hold(c, "off");
```

After creating the data with `PLotLib::getData()` and assigning it to the `PLotLibData` object, we create a new figure and set the plot configuration to always used the recently created figure 1 (lines 7–8). Then, we just need to use `PLotLib::set_hold()` for overlapping all plotting graphics.

Figure 4.2 shows an example with a three-dimensional surface plot. Its code is shown below:

```
1  PlotLib::sendCommand("[X_mesh,Y_mesh]=meshgrid(-2:.4:2)");
2
3  Mat X = PlotLib::getData("X_mesh");
4  Mat Y = PlotLib::getData("Y_mesh");
5  Mat Z = PlotLib::getData("X_mesh.*exp(-X_mesh.^2-Y_mesh.^2)");
6
7  PlotLibData d;
8  d.set_x(X);
9  d.set_y(Y);
10 c.set_z(Z);
11
12 PlotLibConfig c_surf;
13 c.set_xlabel("X-axis");
14 c.set_ylabel("Y-axis");
15 c.set_zlabel("Z-axis");
16
17 PL::surf(d, c);
```

The interesting feature here is that `PlotLib::sendCommand` is used to create variables from a more complex statement, Now, `PlotLib::getData()` only retrieves the already created variables. In figure 4.3 we find the resulting plot for two other three-dimensional functions.

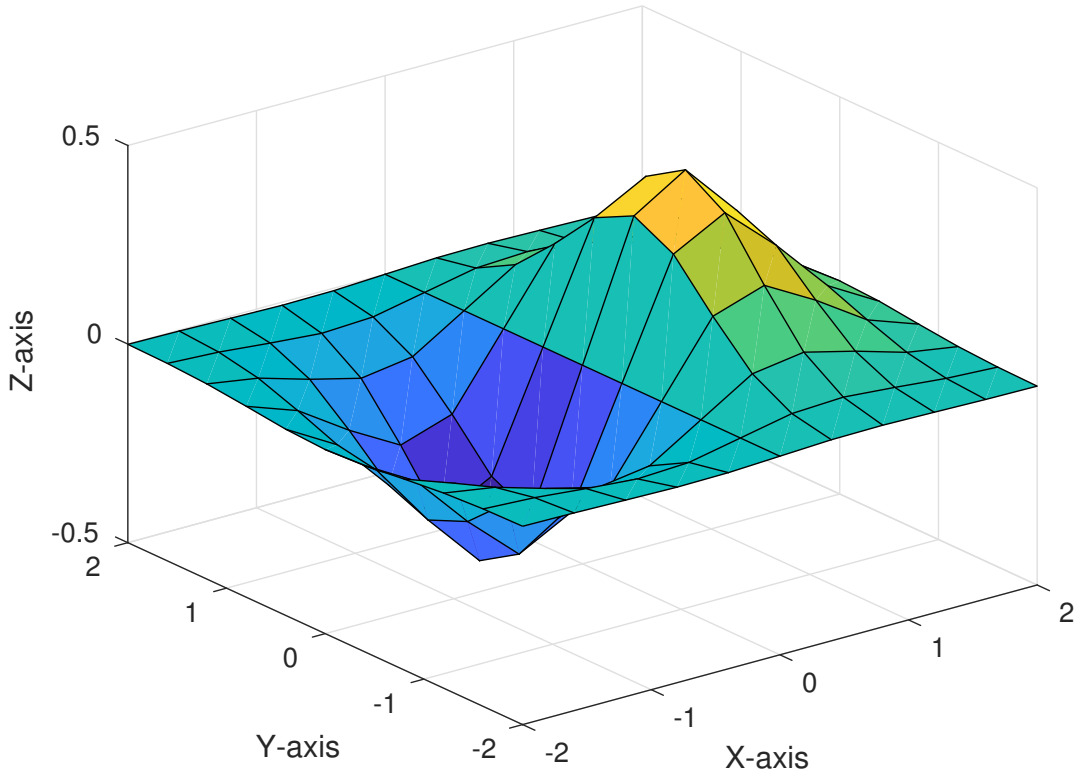
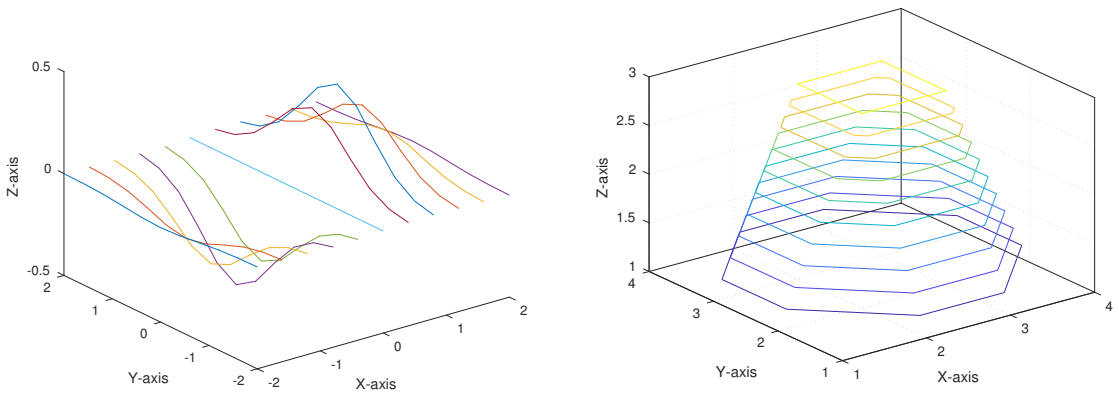


Figure 4.2: Three-dimensional surface plot example



(a) *plot3* function

(b) *contour3* function

Figure 4.3: Other three-dimensional functions

Chapter 5

Licensing

We want to make this library available open-source in the department's gitlab repository. Hence, it was needed to investigate the available options for licensing our work. There are three main options concerning open-source licenses:

- **MIT License:** this is the most permissive option. It is short and lets third-parties use the code in any fashion as long as don't hold you liable and reference the original work.
- **Apache License 2.0:** it works similar as the MIT License. However, it also provides an express grant of patent rights from contributors to users.
- **GNU GPLv3:** the last option is a copyleft license that requires anyone who distributes your code or a derivative work to make the source available under the same terms. Like Apache 2.0, it also covers patent rights.

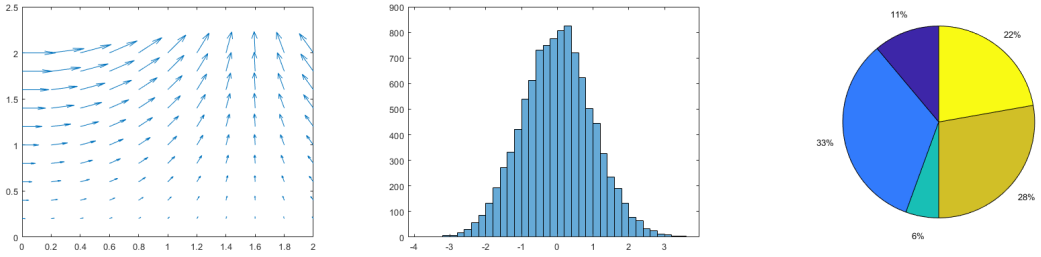
The MIT option was not possible for us because of university's internal policy. Regarding GNU GPLv3, it limits the users of our toolkit to license their code under the same conditions as ours. Because we don't want to constrain the users in such a way, we chose Apache 2.0 to be our license for the project.

Chapter 6

Conclusion

In this research internship, a C++ toolkit for analyzing multimedia algorithms and codecs, especially HEVC, was developed. The plotting data is stored into OpenCV's *cv::Mat* type and structures are available for organizing graphics and figures properties. The library uses a pipe for streaming both data and commands into MATLAB or Octave. Some capabilities include handling of multiple figures, ability to generate OpenCV from simple MATLAB/Octave compliant statements and access to a number of different plotting functions and style configurations. It can be used with simple commands and needs no further configuration from the external engines aside from being installed.

In future work, the library can be expanded to support other available graphic and display functions, such as vector fields, histograms and pie charts (figure 6.1). More-



(a) Vector fields (b) Histograms (c) Pie charts

Figure 6.1: Example of future functions to be implemented

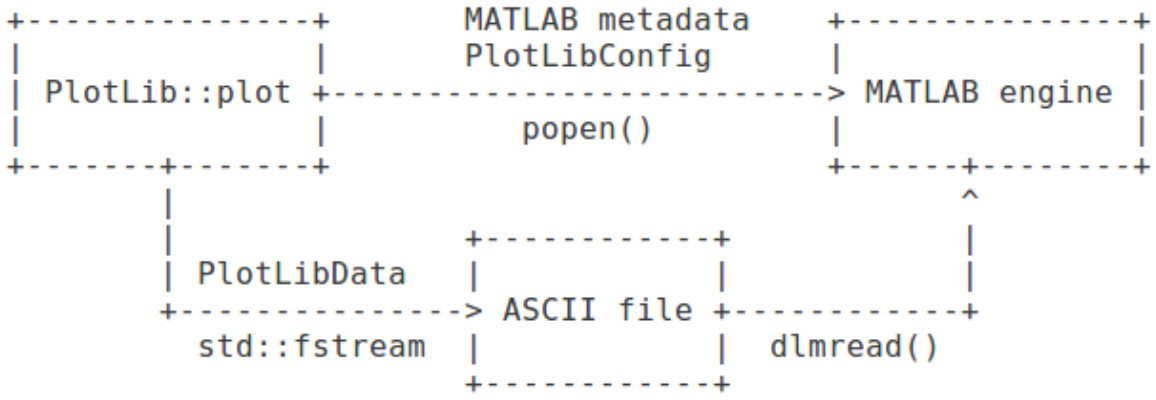


Figure 6.2: File-based data piping

over, interaction between C++ and the external process can be expanded, such as the ability of manipulating variables back and forth.

In spite of this, a task that should be done in the near future regards the plotting data piping. Because `popen()` has an in-built limit for the buffer size, it is not possible to declare matrices or vectors with thousands of points at the external engine. A proposed solution is depicted on figure 6.2. Instead of sending all needed objects throught the opened process, the graphic data contained at the `PlotLibData` structure is saved in an ASCII (text) file. Afterwards, the MATLAB engine is able ro read the data from the text file using the `delmread()` function. This framework is inspired into the implementation `PlotLibData`, which works in a similar way, expect the data stream is done in a reversed fashion with `dlmwrite()`.

Appendix A

Doxygen documentation

In the following pages the autogenerated Doxygen documention for the library is provided. It includes a brief description of every class, function and variable implemented in the code.

hevc-plot

0.24

Generated by Doxygen 1.8.11

Contents

Chapter 1

hevc-plot

Useful links for setting up and using OpenCV:

- http://docs.opencv.org/trunk/d7/d9f/tutorial_linux_install.html
- http://docs.opencv.org/trunk/db/df5/tutorial_linux_gcc_cmake.html
- For building:

```
1 cd build
2 cmake ..
3 make
```

Required libraries:

- OpenCV

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 PlotLib Class Reference

PlotLib class.

`#include <plotlib.h>`

Static Public Member Functions

- static void init ()
- static void close ()
- static std::string readFromProcess ()
- static cv::Mat getData (std::string cmd)
- static bool sendCommand (std::string command)
- static PlotLibConfig twoDimFunc (PlotLibData &d, PlotLibConfig &c, std::string func_name)
- static PlotLibConfig plot (PlotLibData &d, PlotLibConfig c=PlotLibConfig())
- static PlotLibConfig scatter (PlotLibData &d, PlotLibConfig c=PlotLibConfig())
- static PlotLibConfig stem (PlotLibData &d, PlotLibConfig c=PlotLibConfig())
- static PlotLibConfig loglog (PlotLibData &d, PlotLibConfig c=PlotLibConfig())
- static PlotLibConfig semilogx (PlotLibData &d, PlotLibConfig c=PlotLibConfig())
- static PlotLibConfig semilogy (PlotLibData &d, PlotLibConfig c=PlotLibConfig())
- static PlotLibConfig rectangle (cv::Rect r, PlotLibConfig c=PlotLibConfig())
- static PlotLibConfig threeDimFunc (PlotLibData &d, PlotLibConfig &c, std::string func_name)
- static PlotLibConfig contour (PlotLibData &d, PlotLibConfig c=PlotLibConfig())
- static PlotLibConfig plot3 (PlotLibData &d, PlotLibConfig c=PlotLibConfig())
- static PlotLibConfig scatter3 (PlotLibData &d, PlotLibConfig c=PlotLibConfig())
- static PlotLibConfig contour3 (PlotLibData &d, PlotLibConfig c=PlotLibConfig())
- static PlotLibConfig surf (PlotLibData &d, PlotLibConfig c=PlotLibConfig())
- static PlotLibConfig mesh (PlotLibData &d, PlotLibConfig c=PlotLibConfig())
- static PlotLibConfig rectangle3 (cv::Rect r, double z, PlotLibConfig c=PlotLibConfig())
- static void new_figure (std::string name="")
- static void set_hold (PlotLibConfig &c, std::string h)

4.1.1 Detailed Description

PlotLib class.

Class for plotting OpenCV data into octave and MATLAB

4.1.2 Member Function Documentation

4.1.2.1 void PlotLib::close ( ) [static]

Close process

4.1.2.2 PlotLibConfig PlotLib::contour (PlotLibData & *d*, PlotLibConfig *c* = PlotLibConfig()) [static]

Contour plot

Parameters

d	Data strcture.
c	Configuration structure.

Returns

Configuration structure.

4.1.2.3 PlotLibConfig PlotLib::contour3 (PlotLibData & *d*, PlotLibConfig *c* = PlotLibConfig()) [static]

3D Contour plot

Parameters

d	Data strcture.
c	Configuration structure.

Returns

Configuration structure.

4.1.2.4 cv::Mat PlotLib::getData (std::string *cmd*) [static]

Turns MATLAB/Octave data command into cv::Mat

Parameters

cmd	Data command.

Returns

Resulted data from command into cv::Mat

4.1.2.5 void PlotLib::init ( ) [static]

Start piping process

4.1.2.6 PlotLibConfig PlotLib::loglog (PlotLibData & *d,* PlotLibConfig *c* = PlotLibConfig ()) [static]

Log-log scale plot

Parameters

d	Data strcture.
c	Configuration structure.

Returns

Configuration structure.

4.1.2.7 PlotLibConfig PlotLib::mesh (PlotLibData & *d,* PlotLibConfig *c* = PlotLibConfig ()) [static]

Mesh grid plot

Parameters

d	Data strcture.
c	Configuration structure.

Returns

Configuration structure.

4.1.2.8 void PlotLib::new_figure (std::string *name* = " ") [static]

Create figure

Parameters

name	Figure name.

4.1.2.9 PlotLibConfig PlotLib::plot (PlotLibData & *d,* PlotLibConfig *c* = PlotLibConfig ()) [static]

Simple plot

Parameters

d	Data strcture.
c	Configuration structure.

Returns

Configuration structure.

4.1.2.10 PlotLibConfig PlotLib::plot3 (PlotLibData & *d*, PlotLibConfig *c* = PlotLibConfig ()) [static]

3D plot

Parameters

d	Data strcture.
c	Configuration structure.

Returns

Configuration structure.

4.1.2.11 std::string PlotLib::readFromProcess ( ) [static]

Read output from MATLAB/Octave process

Returns

Output from process.

4.1.2.12 PlotLibConfig PlotLib::rectangle (cv::Rect *r*, PlotLibConfig *c* = PlotLibConfig ()) [static]

Draws rectangle in two dimensional space

Parameters

r	Rectangle data.
c	Configuration structure.

Returns

Configuration structure.

4.1.2.13 PlotLibConfig PlotLib::rectangle3 (cv::Rect *r*, double *z*, PlotLibConfig *c* = PlotLibConfig ()) [static]

Draws rectangle in three dimensional space parallel to z-axis

Parameters

r	Rectangle data.
z	Height at z-axis.
c	Configuration structure.

Returns

Configuration structure.

4.1.2.14 PlotLibConfig PlotLib::scatter (PlotLibData & *d,* PlotLibConfig *c* = PlotLibConfig ()) [static]

Scatter plot

Parameters

d	Data strcture.
c	Configuration structure.

Returns

Configuration structure.

4.1.2.15 PlotLibConfig PlotLib::scatter3 (PlotLibData & *d,* PlotLibConfig *c* = PlotLibConfig ()) [static]

3D scatter plot

Parameters

d	Data strcture.
c	Configuration structure.

Returns

Configuration structure.

4.1.2.16 PlotLibConfig PlotLib::semilogx (PlotLibData & *d,* PlotLibConfig *c* = PlotLibConfig ()) [static]

Semilogarithmic plot for x-axis

Parameters

d	Data strcture.
c	Configuration structure.

Returns

Configuration structure.

4.1.2.17 PlotLibConfig PlotLib::semilogy (PlotLibData & *d,* PlotLibConfig *c* = PlotLibConfig ()) [static]

Semilogarithmic plot for y-axis

Parameters

d	Data strcture.
c	Configuration structure.

Returns

Configuration structure.

4.1.2.18 bool PlotLib::sendCommand (std::string *command*) [static]

send command to octave

Parameters

command	Command text.

Returns

If operation was successful.

4.1.2.19 void PlotLib::set_hold (PlotLibConfig & *c*, std::string *h*) [static]

Set figure hold

Parameters

c	Configuration structure.
h	Hold setting.

4.1.2.20 PlotLibConfig PlotLib::stem (PlotLibData & *d*, PlotLibConfig *c* = PlotLibConfig()) [static]

Stem plot

Parameters

d	Data strcture.
c	Configuration structure.

Returns

Configuration structure.

4.1.2.21 PlotLibConfig PlotLib::surf (PlotLibData & *d*, PlotLibConfig *c* = PlotLibConfig ()) [static]

3D Contour plot

Parameters

d	Data strcture.
c	Configuration structure.

Returns

Configuration structure.

4.1.2.22 PlotLibConfig PlotLib::threeDimFunc (PlotLibData & *d*, PlotLibConfig & *c*, std::string *func_name*) [static]

template for three dimensional fuction

Parameters

d	Data strcture.
c	Configuration structure.
func_name	Name of specific function to be called.

Returns

Configuration structure.

4.1.2.23 PlotLibConfig PlotLib::twoDimFunc (PlotLibData & *d*, PlotLibConfig & *c*, std::string *func_name*) [static]

template for two dimensional fuction

Parameters

d	Data strcture.
c	Configuration structure.
func_name	Name of specific function to be called.

Returns

Configuration structure.

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

- /home/pamo/workspace/plotlib/plotlib.h
- /home/pamo/workspace/plotlib/plotlib.cpp

4.2 PlotLibConfig Class Reference

PlotLibConfig class.

`#include <plotlibconfig.h>`

Public Member Functions

- PlotLibConfig ()
- void add_kwargs (std::string key, std::string property)
- bool remove_kwargs (std::string key)
- std::map< std::string, std::string > get_kwargs ()
- void add_ax_conf (std::string key, std::string property)
- bool remove_ax_conf (std::string key)
- std::map< std::string, std::string > get_ax_conf ()
- void set_title (std::string n)
- void set_linespec (std::string f)
- void set_xlabel (std::string xl)
- void set_ylabel (std::string yl)
- void set_zlabel (std::string zl)
- void set_figure (int i)
- std::string get_title ()
- std::string get_linespec ()
- std::string get_xlabel ()
- std::string get_ylabel ()
- std::string get_zlabel ()
- int get_figure ()

4.2.1 Detailed Description

PlotLibConfig class.

Class for handling plot configurations

4.2.2 Constructor & Destructor Documentation

4.2.2.1 PlotLibConfig::PlotLibConfig ( ) [inline]

4.2.3 Member Function Documentation

4.2.3.1 void PlotLibConfig::add_ax_conf (std::string *key*, std::string *property*)

Add axes configuration

Parameters

key	Configuration to add.
property	Property name.

4.2.3.2 void PlotLibConfig::add_kwargs (std::string *key,* std::string *property*)

Add configuration to kwargs

Parameters

key	Configuration to add.
property	Property name.

4.2.3.3 std::map< std::string, std::string > PlotLibConfig::get_ax_conf ( )

Get axes configuration

Returns

Dictionary.

4.2.3.4 int PlotLibConfig::get_figure ( )

get figure ID

Returns

Figure ID.

4.2.3.5 std::map< std::string, std::string > PlotLibConfig::get_kwargs ( )

Get configuration kwargs

Returns

Dictionary.

4.2.3.6 std::string PlotLibConfig::get_linespec ( )

get style code

Returns

Style code.

4.2.3.7 std::string PlotLibConfig::get_title ( )

get plot title

Returns

Plot title.

4.2.3.8 std::string PlotLibConfig::get_xlabel ( )

get X-axis label

Returns

X-axis label.

4.2.3.9 std::string PlotLibConfig::get_ylabel ( )

get Y-axis label

Returns

Y-axis label.

4.2.3.10 std::string PlotLibConfig::get_zlabel ( )

get Z-axis label

Returns

Z-axis label.

4.2.3.11 bool PlotLibConfig::remove_ax_conf (std::string *key*)

Remove axes configuration from kwargs

Parameters

key	Configuration to remove.

Returns

If operation was successful.

4.2.3.12 bool PlotLibConfig::remove_kwargs (std::string *key*)

Remove configuration from kwargs

Parameters

key	Configuration to remove.

Returns

If operation was successful.

4.2.3.13 void PlotLibConfig::set_figure (int *i*)

set figure ID

Parameters

i	Figure ID.

4.2.3.14 void PlotLibConfig::set_linespec (std::string *f*)

set style code

Parameters

f	Style code.

4.2.3.15 void PlotLibConfig::set_title (std::string *n*)

set plot title

Parameters

n	Plot title.

4.2.3.16 void PlotLibConfig::set_xlabel (std::string *xl*)

set X-axis label

Parameters

xl	X-axis label.

4.2.3.17 void PlotLibConfig::set_ylabel (std::string *yl*)

set Y-axis label

Parameters

yl	Y-axis label.

4.2.3.18 void PlotLibConfig::set_zlabel (std::string *zl*)

set Z-axis label

Parameters

zl	Z-axis label.

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

- /home/pamo/workspace/plotlib/plotlibconfig.h
- /home/pamo/workspace/plotlib/plotlibconfig.cpp

4.3 PlotLibData Class Reference

PlotLibData class.

```
#include <plotlibdata.h>
```

Public Member Functions

- void set_x (cv::Mat &x)
- void set_y (cv::Mat &y)
- void set_z (cv::Mat &z)
- cv::Mat * get_x ()
- cv::Mat * get_y ()
- cv::Mat * get_z ()
- bool remove_x ()
- bool remove_y ()
- bool remove_z ()

4.3.1 Detailed Description

PlotLibData class.

Class for handling plot data

4.3.2 Member Function Documentation

4.3.2.1 cv::Mat * PlotLibData::get_x ( )

get x-axis data

Returns

Pointer to data.

4.3.2.2 cv::Mat * PlotLibData::get_y ( )

get y-axis data

Returns

Pointer to data.

4.3.2.3 cv::Mat * PlotLibData::get_z ( )

get z-axis data

Returns

Pointer to data.

4.3.2.4 bool PlotLibData::remove_x ( )

remove x-axis data

Returns

If operation was successful.

4.3.2.5 bool PlotLibData::remove_y ( )

remove y-axis data

Returns

If operation was successful.

4.3.2.6 bool PlotLibData::remove_z ( )

remove z-axis data

Returns

If operation was successful.

4.3.2.7 void PlotLibData::set_x (cv::Mat & *x*)

set x-axis data

Parameters

x	OpenCV data

4.3.2.8 void PlotLibData::set_y (cv::Mat & *y*)

set y-axis data

Parameters

y	OpenCV data

4.3.2.9 void PlotLibData::set_z (cv::Mat & *z*)

set z-axis data

Parameters

z	OpenCV data

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

- /home/pamo/workspace/plotlib/plotlibdata.h
- /home/pamo/workspace/plotlib/plotlibdata.cpp

4.4 PlotLibFigure Class Reference

PlotLibFigure class.

`#include <plotlibfigure.h>`

Public Member Functions

- PlotLibFigure (int i, std::string n)
- std::string get_name ()
- int get_id ()

4.4.1 Detailed Description

PlotLibFigure class.

Class for handling figure objects

4.4.2 Constructor & Destructor Documentation

4.4.2.1 PlotLibFigure::PlotLibFigure (int *i,* std::string *n*)

Class constructor

Parameters

i	Figure ID.
n	Figure name.

4.4.3 Member Function Documentation

4.4.3.1 int PlotLibFigure::get_id ( )

Get figure ID

Returns

Figure ID.

4.4.3.2 std::string PlotLibFigure::get_name ( )

Get figure name

Returns

Figure name.

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

- /home/pamo/workspace/plotlib/plotlibfigure.h
- /home/pamo/workspace/plotlib/plotlibfigure.cpp

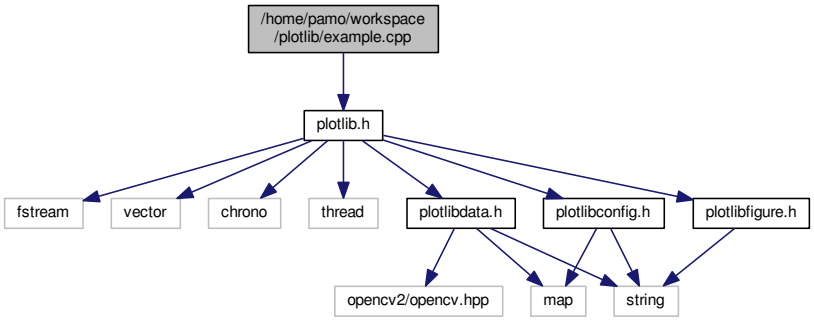
Chapter 5

File Documentation

5.1 /home/pamo/workspace/plotlib/example.cpp File Reference

```
#include "plotlib.h"
```

Include dependency graph for example.cpp:



Functions

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

5.1.1 Function Documentation

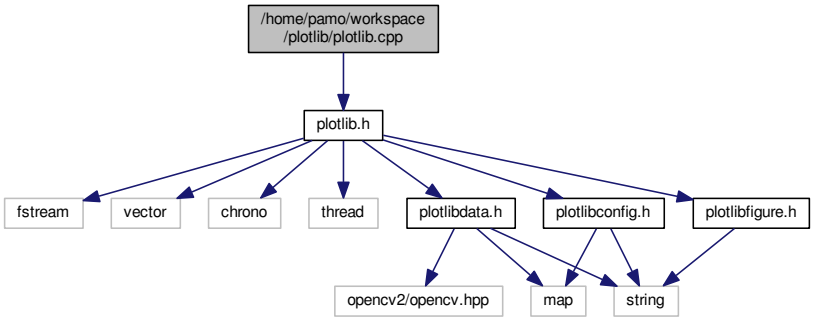
5.1.1.1 int main (int *argc,* char ** *argv*)

5.2 /home/pamo/workspace/plotlib/plotlib.cpp File Reference

Implementation of plotting and engine functions.

```
#include "plotlib.h"
```

Include dependency graph for plotlib.cpp:



5.2.1 Detailed Description

Implementation of plotting and engine functions.

PlotLib Developers (nils.genser@fau.de)

Date

November, 2017

Copyright

Copyright (c) 2017, PlotLib Developers

License

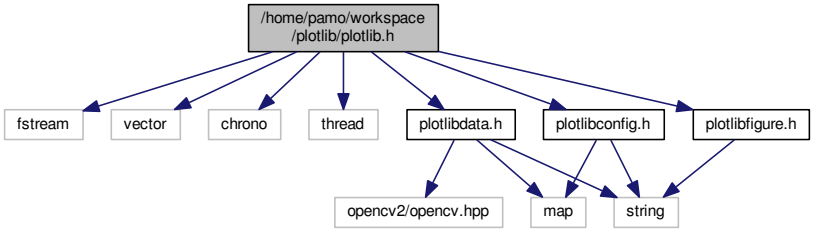
Licensed under the Apache License, Version 2.0

5.3 /home/pamo/workspace/plotlib/plotlib.h File Reference

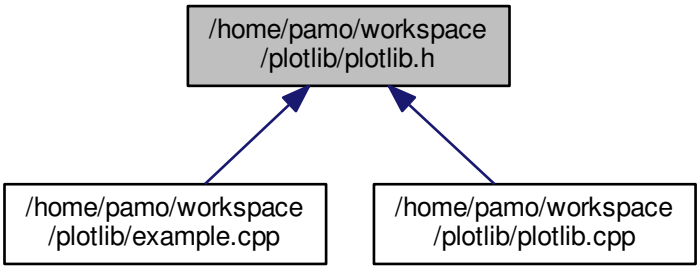
Header for PlotLib class.

```
#include <fstream>
#include <vector>
#include <chrono>
#include <thread>
#include "plotlibdata.h"
#include "plotlibconfig.h"
#include "plotlibfigure.h"
```

Include dependency graph for plotlib.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- class PlotLib

  PlotLib class.

5.3.1 Detailed Description

Header for PlotLib class.

PlotLib Developers (nils.genser@fau.de)

Date

November, 2017

Copyright

Copyright (c) 2017, PlotLib Developers

License

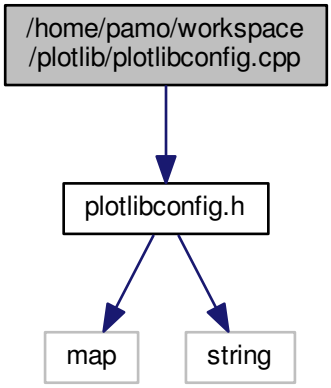
Licensed under the Apache License, Version 2.0

5.4 /home/pamo/workspace/plotlib/plotlibconfig.cpp File Reference

Implementation of configuration handling structure.

```
#include "plotlibconfig.h"
```

Include dependency graph for plotlibconfig.cpp:



5.4.1 Detailed Description

Implementation of configuration handling structure.

PlotLib Developers (nils.genser@fau.de)

Date

November, 2017

Copyright

Copyright (c) 2017, PlotLib Developers

License

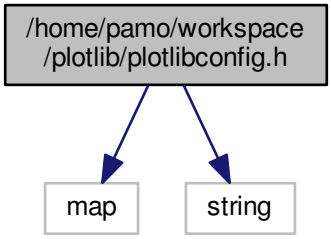
Licensed under the Apache License, Version 2.0

5.5 /home/pamo/workspace/plotlib/plotlibconfig.h File Reference

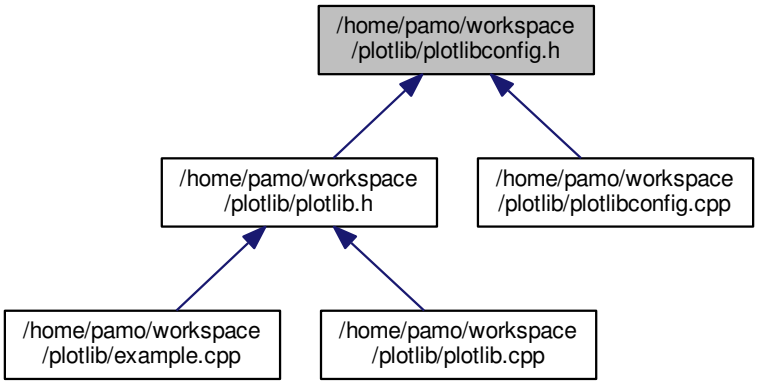
Header for PlotLibConfig class.

```
#include <map>
#include <string>
```

Include dependency graph for plotlibconfig.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- class PlotLibConfig

  PlotLibConfig class.

5.5.1 Detailed Description

Header for PlotLibConfig class.

PlotLib Developers (nils.genser@fau.de)

Date

November, 2017

Copyright

Copyright (c) 2017, PlotLib Developers.

License

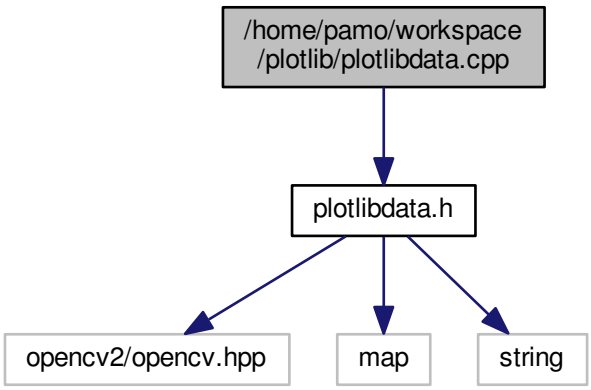
Licensed under the Apache License, Version 2.0

5.6 /home/pamo/workspace/plotlib/plotlibdata.cpp File Reference

Implementation of data handling structure.

```
#include "plotlibdata.h"
```

Include dependency graph for plotlibdata.cpp:



5.6.1 Detailed Description

Implementation of data handling structure.

PlotLib Developers (nils.genser@fau.de)

Date

November, 2017

Copyright

Copyright (c) 2017, PlotLib Developers

License

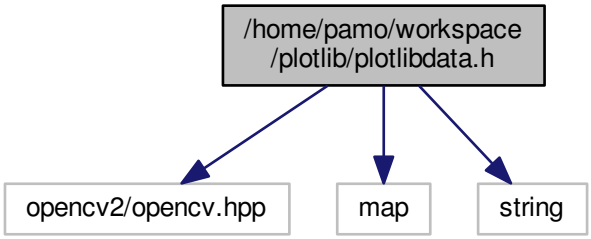
Licensed under the Apache License, Version 2.0

5.7 /home/pamo/workspace/plotlib/plotlibdata.h File Reference

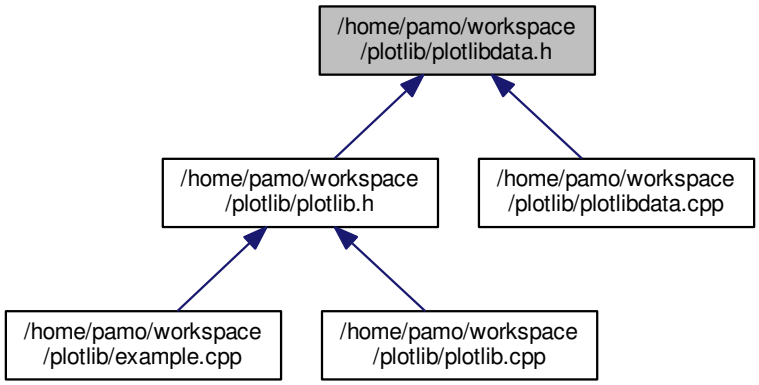
Header for PlotLibData class.

```
#include "opencv2/opencv.hpp"
#include <map>
#include <string>
```

Include dependency graph for plotlibdata.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- class PlotLibData

  PlotLibData class.

5.7.1 Detailed Description

Header for PlotLibData class.

PlotLib Developers (nils.genser@fau.de)

Date

November, 2017

Copyright

Copyright (c) 2017, PlotLib Developers

License

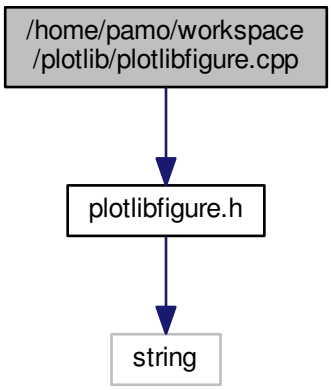
Licensed under the Apache License, Version 2.0

5.8 /home/pamo/workspace/plotlib/plotlibfigure.cpp File Reference

Implementation of figure handling structure.

```
#include "plotlibfigure.h"
```

Include dependency graph for plotlibfigure.cpp:



5.8.1 Detailed Description

Implementation of figure handling structure.

PlotLib Developers (nils.genser@fau.de)

Date

November, 2017

Copyright

Copyright (c) 2017, PlotLib Developers

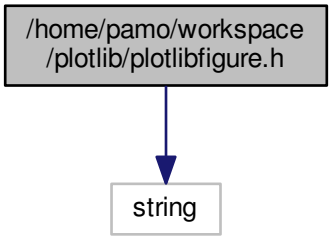
License

Licensed under the Apache License, Version 2.0

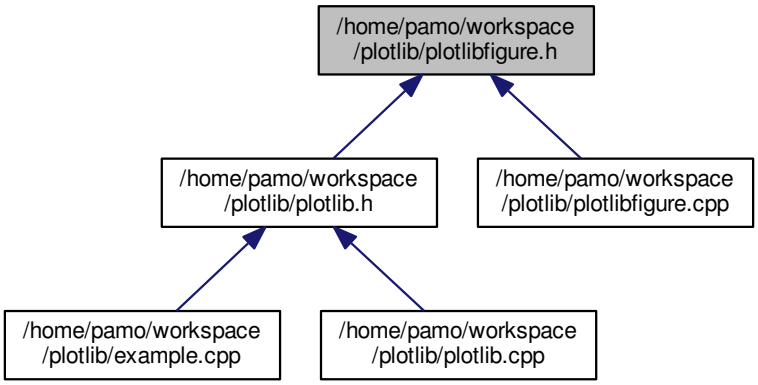
5.9 /home/pamo/workspace/plotlib/plotlibfigure.h File Reference

Header for PlotLibFigure class.

`#include <string>`
Include dependency graph for plotlibfigure.h:



This graph shows which files directly or indirectly include this file:



Data Structures

- class PlotLibFigure
  PlotLibFigure class.

5.9.1 Detailed Description

Header for PlotLibFigure class.

PlotLib Developers (nils.genser@fau.de)

Date

November, 2017

Copyright

License

5.10 /home/pamo/workspace/plotlib/README.md File Reference

Bibliography

[1] Itseez, "OpenCV." `opencv.org`, Nov 2017.

[2] Mathworks Inc., "Matlab." `mathworks.com/products/matlab`, Nov 2017.

[3] GNU Project, "Octave." `gnu.org/software/octave`, Nov 2017.

[4] M. J. LeBrun and G. Furnish, "PLplot." `plplot.sourceforge.net`, Nov 2017.

[5] Max Planck Institute for Solar System Research, "DISLIN." `mps.mpg.de/dislin`, Nov 2017.

[6] B. Evers, "matplotlib-cpp." `github.com/lava/matplotlib-cpp`, Nov 2017.

[7] D. Stahlke, "gnuplot-iostream." `github.com/dstahlke/gnuplot-iostream`, Nov 2017.