colorply

Release 0.0.1

Feb 21, 2021

Getting Started

1 About 1

2 Instalation 3

3 Usage 5

4 Multispectral Photogrammetry 7

5 colorply.mm3d 9

6 colorply.io 13

7 colorply.process 17

8 colorply.ui 21

9 Indices and tables 23

Python Module Index 25

Index 27

CHAPTER 1

About

Colorply is an open source python application which add new wavelength channels to a 3D cloud of points from a set of referenced images and uses images calibration from **MicMac**. This package comes with a GUI, to make tasks easier if you are not used to MicMac.

These models are models oriented in the same relative coordinate system. However, these models are composed of only one channel. **Colorply** merges any channels from any models, as long as they are in the same coordinate system.

This project alone was made in a week, in a bigger project of photogrammetry engineering. The whole project focused in testing and evaluating a multispectral camera, the Parot Sequoia. Then, a major part used this camera to generate multispectral 3D cloud of points for remote sensing purposes like vegetation classification, all in 3D. Because MicMac works only for RGB (or maximum 3-channels images), **Colorply** was created and handles as many channels as you want and complete an existing 3D points from multispectral images.

CHAPTER 2

Instalation

This package depends on :

- PyQt5
- lxml
- plyfile
- numpy

PyQt5 is only used fot the **GUI** of this application. As it can be a bit tricky to understand the file and directory names generated by **MicMac**, we decided to create an interface for it. It is not obligatory to use this interface, you can freely use and modify the python modules.

To install this package, clone this repository and from the command line in the root folder, use :

```
git clone https://github.com/arthurdjn/colorply
cd nets
pip install .
```

Or you can download it directly with pip:

```
pip install colorply
```

This will install all the dependencies and add **Colorply** to your python environment, usually saved in *path_to_anacondalibsite-packages*.

CHAPTER 3

Usage

Colorply works hands in hands with *MicMac* which is an open source photogrammetric software. You can download it from GitHub, or from the main page. If you are new to *MicMac*, take a look a the documentation and this active forum.

3.1 How it works

The interface is simple, but all options are linked !

To use this package, run main.py or

```
from colorply.ui import interface

interface()
```

If you prefer doing it manually, you can use the functional implementation:

```
from colorply import add_cloud_channel

# Load the 3D model
input_ply = "test/data/result/RVB_GRE.ply"
# Load MicMac calibration files
calibration_file = "test/data/calibration/Ori-1bande_All_CampariGCP/AutoCal_Foc-4000_
↪Cam-SequoiaSequoia-GRE.xml"
# Load the orientation files for all images in the scene
orientation_dir = "test/data/calibration/Ori-1bande_All_CampariGCP"
# Load the images (corresponding to the orientation files)
image_dir = "test/data/images/RED"
# Additional arguments
image_ext = "TIF"
channel = "RED"
# Resulting ply
```

(continues on next page)

(continued from previous page)

```
output_ply = "output.ply"

# Project all points in `input_ply` to `images_dir`.
# Create a new ply, the operation is not inplace.
add_cloud_channel(input_ply, output_ply,
                  calibration_file, orientation_dir,
                  image_dir, image_ext, channel)
```

You can also use the commandlines system

```
python colorply     --inply path/to/input_ply
                    --outply path/to/output_ply
                    --calib path/to/calib_xml
                    --oridir path/to/orientation_folder
                    --imdir path/to/image_folder
                    --imext image_extension
                    --channel name_of_new_channel
                    --mode mode_used_to_merge_new_radiometry
```

Use help –argument for additional information.

CHAPTER 4

Multispectral Photogrammetry

This project was used for remote sensing classification from a multispectral camera. The camera *Parrot Sequoia* was first calibrated and tested on sample areas. Then, it was fixed to a civil drone and we flew over high altitudes forest to estimate the evolution of vegetation species. **Colorply** was used to create a multispectral cloud of points, to improve our classification by adding extra features. The clusters were made from a random forest skeleton, using all radiometries (RED, REG, NIR, GRE) and the points 3D positions.

In this repository, you can run the classification on the provided data [here](test/data/result/RVB_GRE_RED_REG_NIR_NDVI.ply). For this short example (for fast processing), the classification results are described as follows :

The confusion matrix for this sample is :

	Terrain	**Oak**	**Shrub**	**Grass**
Terrain	**410**	0	0	16
Oak	0	**260**	10	0
Shrub	0	10	**137**	16
Grass	23	0	11	**192**

Global accuracy : 92.07%.

CHAPTER 5

colorply.mm3d

5.1 colorply.mm3d.calibration

Class object for calibration definition. Note that class objects are heavy weight in python, therefore these class won't be (always) used in the main core of colorply.

class colorply.mm3d.calibration.**Calib**(*filename*)
A Calib object stores camera calibration information generated by MicMac. The calibration have always a xml extension - a MicMac standard.

class colorply.mm3d.calibration.**Ori**(*filename*)
An Ori object stores orientation information generated by MicMac. The orientation files have always a xml extension - a MicMac standard.

5.2 colorply.mm3d.functional

Read and extract informations from MicMac xml calibration files. Functional implementation.

colorply.mm3d.functional.**read_S**(*nameIMGxml*)
This function extracts the images's center from the xml file.

Note: Usually, it is "Orientation-Im[n°i].JPG.xml"

Parameters **nameIMGxml** (*str*) – The name of the file generated by MM3D.

Returns Center of the IMG, of shape 1x3

Return type np.ndarray

colorply.mm3d.functional.**read_calib**(*calibxml*)
This function extracts the calibration parameters from the xml file.

Note: Usually, it is similar to "AutoCal_[Focal]_[CameraName].xml"

Parameters **calibxml** (*str*) – Name of the camera calibration file

Returns Returns F, PPS, distorsion coefficients a, b, c, size

Return type dict

colorply.mm3d.functional.**read_calib_F**(*calibxml*)
This function extracts the calibration parameters from the xml file.

Note: Usually, it is similar to "AutoCal_[Focal]_[CameraName].xml"

Parameters **calibxml** (*str*) – Name of the camera calibration file

Returns Coordinates of the point F (focale, units : pix)

Return type np.ndarray

colorply.mm3d.functional.**read_calib_PPS**(*calibxml*)
This function extracts the calibration parameters from the xml file.

Note: Usually, it is similar to "AutoCal_[Focal]_[CameraName].xml"

Parameters **calibxml** (*str*) – Name of the camera calibration file

Returns Coordinates of the PPS, of shape 1x3

Return type np.ndarray

colorply.mm3d.functional.**read_calib_distorsion**(*calibxml*)
This function extracts the calibration parameters from the xml file.

Note: Usually, it is similar to "AutoCal_[Focal]_[CameraName].xml"

Parameters **calibxml** (*str*) – Name of the camera calibration file

Returns Distorsion coefficients [a, b, c]

Return type np.ndarray

colorply.mm3d.functional.**read_ori**(*nameIMGxml*)
This function extracts the rotation matrix from the xml file and extracts the images's center from the xml file.

Note: Usually, it is "Orientation-Im[n°i].JPG.xml"

Parameters **nameIMGxml** (*str*) – The name of the file generated by MM3D.

Returns The rotation of the img, the center of the IMG, tuple(np.array(matrix rotation), np.array(coord S))

Return type tuple

colorply.mm3d.functional.**read_orientation**(*nameIMGxml*)
This function extracts the rotation matrix from the xml file.

Parameters

- **nameIMGxml** (*str*) – name of the file generated by MM3D.
- **note::** (.) – Usually, it is "Orientation-Im[n°i].JPG.xml"

Returns Rotation of the img, of shape 3x3

Return type np.ndarray

colorply.mm3d.functional.**read_size**(*calibxml*)
This function extracts the size of an image from the xml file.

Note: Usually, it is similar to "AutoCal_[Focal]_[CameraName].xml"

Parameters **calibxml** (*str*) – Name of the camera calibration file

Returns

- *np.ndarray*
- *The size of the image resolution*

CHAPTER 6

colorply.io

6.1 colorply.io.imfile

Load a set of images with their orientation.

colorply.io.imfile.**load_images**(*orientation_dir*, *image_dir='.'*, *image_ext='TIF'*, *channel='unk'*)

Reads all images and returns a list of image objects

Parameters

- **orientation_dir** (*str*) – orientation directory of MicMac files
- **image_dir** (*str*) – image directory
- **image_ext** (*str*) – image extensiton (ex: 'TIFF')
- **channel** (*str*) – name of the channel

Returns

Return type list(*Image*)

6.2 colorply.io.ply

Contains reading and writing functions for ply files as well as a conversion function and other useful functions related to ply files

colorply.io.ply.**add_channel_from_plydata**(*plydata*, *coord*, *channel*)

Add a channel to the numpy data.

Parameters

- **plydata** (*plyfile.PlyData*) – Raw data.
- **coord** (*numpy.ndarray*) – The coordinates of the 3D points.

- **channel** (*str*) – Name of the channel to add.

Returns **data** – DESCRIPTION.

Return type TYPE

colorply.io.ply.**extract_channel_from_plydata**(*plydata*, *channel='all'*)
Extract a channel from a 3D cloud points.

Parameters

- **plydata** (*plyfile.PlyData*) – The data to extract the channel.
- **channel** (*str, optional*) – The name of the channel to extract. The default is "all".

Returns **data** – The extracted channel data.

Return type numpy.ndaray

colorply.io.ply.**plydata_to_array**(*plydata*)
Convert a plydata to numpy array.

Parameters **plydata** (*plyfile.PlyData*) – The plydata to convert.

Returns The converted data, containing the 3D coordinates of the plydata's points.

Return type numpy.ndarray

colorply.io.ply.**read_plyfile**(*file_name*)
Read and extract the data of a ply file.

Parameters **file_name** (*str*) – The path to the ply file.

Raises FileNotFoundError – If the file is not found, raises an error.

Returns **plydata** – The data of the cloud points file.

Return type plyfile.PlyData

colorply.io.ply.**write_plydata**(*plydata*, *data_channel*, *new_channel_name*, *outfile_name='my_cloud.ply'*)
Create a ply file from plydata. Can add an additional channel to the plydata.

Parameters

- **plydata** (*plyfile.PlyData*) – The plydata to write in the file.
- **data_channel** (*numpy.ndarray*) – Data of the added channel.
- **new_channel_name** (*str*) – Name of the new added channel.
- **outfile_name** (*str, optionnal*) – Name of the ply file created. The default is "my_cloud.ply".

Returns

Return type None

6.3 colorply.io.readxml

colorply.io.readxml.**read_S**(*nameIMGxml*)
This function extract the images's center from the xml file.

Parameters **nameIMGxml** (*str*) – the name of the file generated by MM3D. Usually, it is "Orientation-Im[n°i].JPG.xml"

Returns numpy.ndarray

Return type the center of the IMG (size 1*3)

colorply.io.readxml.**read_calib**(*calibxml*)
This function extract the calibration parameters from the xml file.

Parameters **calibxml** (*str*) – the name of the calibration file generated by MM3D (ex: "AutoCal_Foc-24000_Cam-DSLRA850.xml")

Returns tuple

Return type F, PPS, distorsion coefficients a, b, c, size

colorply.io.readxml.**read_calib_F**(*calibxml*)
This function extract the calibration parameters from the xml file.

Parameters **calibxml** (*str*) – the name of the calibration file generated by MM3D. (ex: "AutoCal_Foc-24000_Cam-DSLRA850.xml")

Returns numpy.ndarray

Return type coordinates of the point F (focale, units : pix), (size 1*)

colorply.io.readxml.**read_calib_PPS**(*calibxml*)
This function extract the calibration parameters from the xml file.

Parameters **calibxml** (*str*) – the name of the calibration file generated by MM3D (ex: "AutoCal_Foc-24000_Cam-DSLRA850.xml").

Returns numpy.ndarray

Return type coordinates of the PPS (size 1*3)

colorply.io.readxml.**read_calib_distorsion**(*calibxml*)
This function extract the calibration parameters from the xml file.

Parameters **calibxml** (*str*) – the name of the calibration file generated by MM3D (ex: "AutoCal_Foc-24000_Cam-DSLRA850.xml").

Returns numpy.ndarray

Return type distorsion coefficients a, b, c (size 1*3)

colorply.io.readxml.**read_ori**(*nameIMGxml*)
This function extract the rotation matrix from the xml file and extract the images's center from the xml file.

Parameters **nameIMGxml** (*str*) – the name of the file generated by MM3D. Usually, it is "Orientation-Im[n°i].JPG.xml"

Returns tuple

Return type the rotation of the img, the center of the IMG, (matrix rotation, coord S)

colorply.io.readxml.**read_orientation**(*nameIMGxml*)
This function extract the rotation matrix from the xml file.

Parameters **nameIMGxml** (*str*) – the name of the file generated by MM3D. Usually, it is "Orientation-Im[n°i].JPG.xml"

Returns numpy.ndarray

Return type the rotation of the img (size 3*3)

colorply.io.readxml.**read_size**(*calibxml*)
This function extract the size of an image from the xml file.

Parameters **calibxml** (*str*) – the name of the calibration file generated by MM3D.

Returns **numpy.ndarray**

Return type the size of the image resolution

CHAPTER 7

colorply.process

7.1 colorply.process.image

Defines an Image, with its orientation, focal, symmetry.

class colorply.process.image.**Image**(*name='None'*, *channel='None'*, *data=<sphinx.ext.autodoc.importer._MockObject object>*, *R=<sphinx.ext.autodoc.importer._MockObject object>*, *S=<sphinx.ext.autodoc.importer._MockObject object>*, *size=(4000, 3000)*)

Define an image, with its name, channels, data, rotation and autocollimation.

7.2 colorply.process.imformula

This modules contains the necessary functions to compute the image formula for each "ground" point.

colorply.process.imformula.**image_formula**(*F*, *M*, *R*, *S*)

Compute the image formula for the point M withoud distorsion

Parameters

- **F** (*numpy.ndarray*) – Position of the autocollimation point in the image coordinate system.
- **M** (*numpy.ndarray*) – Position of the point in real space coordinates.
- **R** (*numpy.ndarray*) – Rotation matrix representing the orientation of the image coordinate system in the real space coordinate system.
- **S** (*numpy.ndarray*) – Position of the autocollimation point in the real space coordinate system.

Returns Image coordinates of M projected.

Return type numpy.ndarray

colorply.process.imformula.**image_formula_corrected**(*F*, *M*, *R*, *S*, *pps*, *a*, *b*, *c*)
Compute the image formula for the point M with distorsion.

Parameters

- **F** (*numpy.ndarray*) – Position of the autocollimation point in the image coordinate system.
- **M** (*numpy.ndarray*) – Position of the point in real space coordinates.
- **R** (*numpy.ndarray*) – Rotation matrix representing the orientation of the image coordinate system in the real space coordinate system.
- **S** (*numpy.ndarray*) – Position of the autocollimation point in the real space coordinate system.
- **pps** (*numpy.ndarray*) – Position of the point of 0 distorsion ine the radial_std model.
- **a** (*float*) – 3rd order coefficient of the distorsion polynomial.
- **b** (*float*) – 5th order coefficient of the distorsion polynomial.
- **c** (*float*) – 7th order coefficient of the distorsion polynomial.

Returns Corrected point position.

Return type numpy.ndarray

colorply.process.imformula.**radial_std**(*m_image*, *pps*, *a*, *b*, *c*)
Corrects the postion of the point according to the standard radial distorsion model.

Note: We use Horner's method to evaluate $ar^2 + br^4 + cr^6$

Parameters

- **m_image** (*numpy.ndarray*) – Position of the projected point in pixel.
- **pps** (*numpy.ndarray*) – Position of the point of 0 distorsion in the radial model.
- **a** (*float*) – 3rd order coefficient of the distorsion polynomial.
- **b** (*float*) – 5th order coefficient of the distorsion polynomial.
- **c** (*float*) – 7th order coefficient of the distorsion polynomial.

Returns Corrected point position.

Return type numpy.ndarray

7.3 colorply.process.improcess

Module to preprocess images.

colorply.process.improcess.**add_cloud_channel**(*input_ply*, *output_ply*, *calibration_file*, *orientation_dir*, *image_dir*, *image_ext='TIF'*, *channel='unk'*, *mode='avg'*, *progress=None*)
All together. Project all points from a ply file.

Parameters

- **input_ply** (*plydata*) – The cloud points to add a new channel.

- **output_ply** (*plydata*) – The output cloud points, with the new channel.
- **calibration_file** (*str*) – Path to the MicMac calibration file.
- **orientation_dir** (*str*) – Path to the MicMac images orientation directory.
- **image_dir** (*str*) – Path to the images with the channel to add.
- **image_ext** (*str, optional*) – Images extension (JPG, TIFF, PNG etc.). The default is "TIF".
- **channel** (*str, optional*) – Channe name. The default is "unk".
- **mode** (*str, optional*) – Way to add the new radiometry to the cloud points. The default is "avg".
- **progress** (*PyQt progress bar, optional*) – The bar of progress. The default is None.

Raises FileNotFoundError – If files are not found, raise an error.

Returns Return True when done.

Return type Bool

colorply.process.improcess.**radiometry_projection**(*M*, *images_loaded*, *calibration*, *mode='avg'*, *scale=0.0038909912109375*)

This function add to a point M a new channel, computed from the loaded images. Therefore, the images should be calibrated in the same reference of your point M. Usually, the 3D point M is part of a cloud points. The point M is projected in all images that see the point. Then, the radiometry from the images channel is added to the point, with different mode.

Parameters

- **M** (*numpy.ndarray*) – Position of the point in real space coordinates.
- **images_loaded** (*list of Images*) – List of the image loaded. These images need to be referenced in the same system as the point M. Usually with MicMac calibrate all the images together.
- **calibration** (*dict*) – dict containing the camera calibration global parameters.
- **mode** (*str, optional*) – The way the new radiometry is stacked in the new M channel. This can be with a mean of all radiometry that see the 3D point M. The default is "avg".
- **scale** (*float*) – Used to scale a channel to [0, 255] values. For Sequoia camera, use a scale factor of 0.0038909912109375.

Returns Value of the new channel.

Return type float

CHAPTER 8

colorply.ui

8.1 colorply.ui.call

This module recieves instruction from the ui and call the appropriate functions on the appropriate parameters.

8.2 colorply.ui.gui

This is the python-colorply GUI.

class colorply.ui.gui.**MainWindow**

compute()
Run the process module, with different threads.

Returns

Return type None.

initUI()
Initialyze the window with different buttons and actions.

Returns

Return type None.

select_calib_dir()
Select the MicMac calibration directory from the window.

Returns

Return type None.

select_image_dir()
Select the image directory from the window.

Returns

Return type None.

select_input_ply()
Select the input ply file from the window.

Returns

Return type None.

select_ori_dir()
Select the MicMac orientation directory from the window.

Returns

Return type None.

select_output_ply()
Select the output ply file from the window.

Returns

Return type None.

class colorply.ui.gui.**RunThread**

run(*window*)
Run the process in a different thread.

Parameters **window** (*PyQT5 window*) – The main window.

Returns

Return type None.

colorply.ui.gui.**interface**()
Create the main window of Colorply.

Returns

Return type None.

8.3 colorply.ui.palette

Set colors and theme to the interface.

colorply.ui.palette.**set_dark_theme**(*application*)
Set a darker theme on the PyQt window.

Parameters **application** (*PyQt Application*) – The application to change theme.

Returns **app** – The new application an theme.

Return type PyQt Application

CHAPTER 9

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

colorply.io.imfile, 13
colorply.io.ply, 13
colorply.io.readxml, 14
colorply.mm3d.calibration, 9
colorply.mm3d.functional, 9
colorply.process.image, 17
colorply.process.imformula, 17
colorply.process.improcess, 18
colorply.ui.call, 21
colorply.ui.gui, 21
colorply.ui.palette, 22

Index

A

add_channel_from_plydata() (*in module colorply.io.ply*), 13
add_cloud_channel() (*in module colorply.process.improcess*), 18

C

Calib (*class in colorply.mm3d.calibration*), 9
colorply.io.imfile (*module*), 13
colorply.io.ply (*module*), 13
colorply.io.readxml (*module*), 14
colorply.mm3d.calibration (*module*), 9
colorply.mm3d.functional (*module*), 9
colorply.process.image (*module*), 17
colorply.process.imformula (*module*), 17
colorply.process.improcess (*module*), 18
colorply.ui.call (*module*), 21
colorply.ui.gui (*module*), 21
colorply.ui.palette (*module*), 22
compute() (*colorply.ui.gui.MainWindow method*), 21

E

extract_channel_from_plydata() (*in module colorply.io.ply*), 14

I

Image (*class in colorply.process.image*), 17
image_formula() (*in module colorply.process.imformula*), 17
image_formula_corrected() (*in module colorply.process.imformula*), 17
initUI() (*colorply.ui.gui.MainWindow method*), 21
interface() (*in module colorply.ui.gui*), 22

L

load_images() (*in module colorply.io.imfile*), 13

M

MainWindow (*class in colorply.ui.gui*), 21

O

Ori (*class in colorply.mm3d.calibration*), 9

P

plydata_to_array() (*in module colorply.io.ply*), 14

R

radial_std() (*in module colorply.process.imformula*), 18
radiometry_projection() (*in module colorply.process.improcess*), 19
read_calib() (*in module colorply.io.readxml*), 15
read_calib() (*in module colorply.mm3d.functional*), 9
read_calib_distorsion() (*in module colorply.io.readxml*), 15
read_calib_distorsion() (*in module colorply.mm3d.functional*), 10
read_calib_F() (*in module colorply.io.readxml*), 15
read_calib_F() (*in module colorply.mm3d.functional*), 10
read_calib_PPS() (*in module colorply.io.readxml*), 15
read_calib_PPS() (*in module colorply.mm3d.functional*), 10
read_ori() (*in module colorply.io.readxml*), 15
read_ori() (*in module colorply.mm3d.functional*), 10
read_orientation() (*in module colorply.io.readxml*), 15
read_orientation() (*in module colorply.mm3d.functional*), 11
read_plyfile() (*in module colorply.io.ply*), 14
read_S() (*in module colorply.io.readxml*), 14
read_S() (*in module colorply.mm3d.functional*), 9
read_size() (*in module colorply.io.readxml*), 15
read_size() (*in module colorply.mm3d.functional*), 11
run() (*colorply.ui.gui.RunThread method*), 22

RunThread (*class in colorply.ui.gui*), 22

S

select_calib_dir() (*colorply.ui.gui.MainWindow method*), 21
select_image_dir() (*colorply.ui.gui.MainWindow method*), 21
select_input_ply() (*colorply.ui.gui.MainWindow method*), 22
select_ori_dir() (*colorply.ui.gui.MainWindow method*), 22
select_output_ply() (*colorply.ui.gui.MainWindow method*), 22
set_dark_theme() (*in module colorply.ui.palette*), 22

W

write_plydata() (*in module colorply.io.ply*), 14