Pycmac Documentation

Release 0.1

Ciaran Robb

Sep 17, 2019

CONTENTS

pycmac is a Python module for using the Micmac Structure from Motion (SfM) library within a python environment.

The module also contains various fuctionality for manipulating data associated with the SfM process as well as some enhancements to the basic micmac routines, such as processing Micasense multi-spectral data.

The aim is to produce convenient, minimal commands for putting together SfM workflows using python and the excellent MicMac lib.

CHAPTER

ONE

CONTENTS

1.1 Quickstart

1.1.1 Notes

Be sure to assign paths with paths to your own data for the folder (where your images are) and csv variables where appropriate.

1.1.2 A workflow using the Malt algorithm

The following simple example uses the pycmac modules

```
from pycmac import orientation, dense_match
```

Perform the relative orientation of images (poses).

```
folder = path/to/my/folder

orientation. feature_match(folder, proj="30 +north",  ext="JPG", schnaps=True)
```

Perform the bundle adjustment with GPS information.

```
orientation.bundle_adjust(folder, algo="Fraser", proj="30 +north",
                  ext="JPG", calib="pathtocsv.csv", gpsAcc='1')
```

Perform the dense matching using the malt algorithm. The args for the dense matching algorithms are largely identical to the MicMac commands (Malt & PIMs), but carry out additional masking, georeferencing and subsetting.

```
dense_match.malt(folder, proj="30 +north", mode='Ortho', ext="JPG", orientation=
↪"Ground_UTM",
         DoOrtho='1',  DefCor='0')
```

Mosaicing can be performed using Tawny or seamline-feathering (enhanced to process multi-band) and ossim.

The examples below are Tawny and seamline-feathering.

Please note that seamline-feathering for multi-band imagery (including RGB) the “ms” variable must be specified below. If not, it will return a greyscale mosaic.

```
dense_match.tawny(folder, proj="30 +north", mode='Malt')
```

(continues on next page)

(continued from previous page)

```
dense_match.feather(folder, proj="ESPG:32360", mode='Malt', ApplyRE="1", ms=['r', 'g',
↪ 'b'])
```

1.2 pycmac

1.2.1 pycmac package

Module contents

pycmac.orientation module

Created on Wed Jun 12 13:54:55 2019

@author: Ciaran Robb

A module which calls Micmac dense matching commands

This is just for convenince to keep everything in python - MicMac has an excellent command line

https://github.com/Ciaran1981/Sfm

orientation.**bundle_adjust**(*folder*, *algo='Fraser'*, *csv=None*, *proj='30 +north'*, *ext='JPG'*, *calib=None*, *SH='_mini'*, *gpsAcc='1'*, *exif=False*)

A function running the relative orientation/bundle adjustment with micmac

Notes

Purely for convenience within python - not necessary - the mm3d cmd line is perfectly good

Parameters

- **folder** (*string*) – working directory
- **proj** (*string*) – a UTM zone eg "30 +north"
- **csv** (*string*) – a csv file of image coordinates in micmac format for a calibration subset needed regardless of whether or not the exif has GPS embedded
- **calib** (*string*) – a calibration subset (optional)
- **ext** (*string*) – image extention e.g JPG, tif
- **SH** (*string*) – a reduced set of tie points (output of schnaps command)
- **gpsAcc** (*string*) – an estimate in metres of the onboard GPS accuracy
- **exif** (*bool*) – if the GPS info is embedded in the image exif check this as True to convert back to geographic coordinates, If previous steps always used a csv for img coords ignore this

orientation.**feature_match**(*folder*, *csv=None*, *proj='30 +north'*, *resize=None*, *ext='JPG'*, *schnaps=True*)

A function running the feature detection and matching with micmac

Notes

Purely for convenience within python - not necessary - the mm3d cmd line is perfectly good

Parameters

- **folder** (*string*) – working directory
- **proj** (*string*) – a UTM zone eg "30 +north"
- **resize** (*string*) – The long axis in pixels to optionally resize the imagery
- **ext** (*string*) – image extention e.g JPG, tif

pycmac.dense_match module

Created on Mon Jun 10 10:52:56 2019

@author: Ciaran Robb

A module which calls Micmac dense matching commands

This is just for convenince to keep everything in python - MicMac has an excellent command line

It does also georef the files

https://github.com/Ciaran1981/Sfm

@author: ciaran

dense_match.**Malt**(*folder*, *proj='30 +north'*, *mode='Ortho'*, *ext='JPG'*, *orientation='Ground_UTM'*, *DoOrtho='1'*, *DefCor='0'*, ***kwargs*)

A function calling the Malt command for use in python

Notes

Purely for convenience within python - not necessary - the mm3d cmd line is perfectly good

see MicMac tools link for further possible kwargs - just put the module cmd as a kwarg The kwargs must be exactly the same case as the mm3d cmd options e.g = UseGpu='1'

Parameters

- **folder** (*string*) – working directory
- **proj** (*string*) – a UTM zone eg "30 +north"
- **mode** (*string*) – Correlation mode - Ortho, UrbanMNE, GeomImage
- **ext** (*string*) – image extention e.g JPG, tif
- **orientation** (*string*) – orientation folder to use (generated by previous tools/cmds) default is "Ground_UTM"

dense_match.**PIMs**(*folder*, *mode='BigMac'*, *ext='JPG'*, *orientation='Ground_UTM'*, *DefCor='0'*, ***kwargs*)

A function calling the PIMs command for use in python

Notes

Purely for convenience within python - not necessary - the mm3d cmd line is perfectly good

see MicMac tools link for further possible args - just put the module cmd as a kwarg The kwargs must be exactly the same case as the mm3d cmd options

Parameters

- **folder** (*string*) – working directory
- **proj** (*string*) – a proj4/gdal like projection information e.g ESPG:32360
- **mode** (*string*) – Correlation mode - MicMac, BigMac, QuickMac, Forest, Statue Default is BigMac
- **ext** (*string*) – image extention e.g JPG, tif
- **orientation** (*string*) – orientation folder to use (generated by previous tools/cmds) default is "Ground_UTM"
- **e.g = UseGpu='1'**

dense_match.**PIMs2MNT**(*folder*, *proj='30 +north'*, *mode='BigMac'*, *DoOrtho='1'*, ***kwargs*)
A function calling the PIMs2MNT command for use in python

Notes

Purely for convenience within python - not necessary - the mm3d cmd line is perfectly good

see MicMac tools link for further possible args - just put the module cmd as a kwarg The kwargs must be exactly the same case as the mm3d cmd options

Parameters

- **folder** (*string*) – working directory
- **proj** (*string*) – a proj4/gdal like projection information e.g "ESPG:32360"
- **mode** (*string*) – Correlation folder to grid/rectify - MicMac, BigMac, QuickMac, Forest, Statue Default is BigMac

dense_match.**Tawny**(*folder*, *proj='30 +north'*, *mode='PIMs'*, ***kwargs*)
A function calling the PIMs2MNT command for use in python

Notes

Purely for convenience within python - not necessary - the mm3d cmd line is perfectly good

see MicMac tools link for further possible args - just put the module cmd as a kwarg The kwargs must be exactly the same case as the mm3d cmd options

Parameters

- **folder** (*string*) – working directory
- **proj** (*string*) – a proj4/gdal like projection information e.g "ESPG:32360"
- **mode** (*string*) – Correlation folder to grid/rectify - MicMac, BigMac, QuickMac, Forest, Statue Default is BigMac

pycmac.utilities module

Created on Tue May 29 16:20:58 2018

@author: ciaran

calib_subset.py -folder mydir -algo Fraser -csv mycsv.csv

utilities.**calib_subset**(*folder*, *csv*, *ext='JPG'*, *algo='Fraser'*)
A function for calibrating on an image subset then initialising a global orientation

Notes

see MicMac tools link for further possible kwargs - just put the module cmd as a kwarg

Parameters

- **folder** (*string*) – working directory
- **proj** (*string*) – a UTM zone eg "30 +north"
- **mode** (*string*) – Correlation mode - Ortho, UrbanMNE, GeomImage
- **ext** (*string*) – image extention e.g JPG, tif
- **orientation** (*string*) – orientation folder to use (generated by previous tools/cmds) default is "Ground_UTM"

utilities.**convert_c3p**(*folder*, *lognm*, *ext='JPG'*)
Edit csv file for c3p to work with MicMac.

This is intended for the output from the software of a C-Astral drone

This assumes the column order is name, x, y, z

Parameters

- **folder** (*string*) – path to folder containing jpegs
- **lognm** (*string*) – path to c3p derived csv file

utilities.**make_sys_utm**(*folder*, *proj*)

utilities.**make_xml**(*csvFile*, *folder*, *yaw=None*)
Make an xml based for the rtl system in micmac

Parameters **csvFile** (*string*) – csv file with coords to use

utilities.**mask_raster_multi**(*inputIm*, *mval=1*, *outval=None*, *mask=None*, *blocksize=256*, *FMT=None*, *dtype=None*)
Perform a numpy masking operation on a raster where all values corresponding to mask value are retained - does this in blocks for efficiency on larger rasters

Parameters

- **inputIm** (*string*) – the input raster
- **mval** (*int*) – the masking value that delineates pixels to be kept
- **outval** (*numerical dtype eg int, float*) – the areas removed will be written to this value default is 0
- **mask** (*string*) – the mask raster to be used (optional)
- **FMT** (*string*) – the output gdal format eg 'Gtiff', 'KEA', 'HFA'

blocksize [int] the chunk of raster read in & write out

utilities.**mv_subset**(*csv*, *inFolder*, *outfolder*)
Move a subset of images based on a MicMac csv file

Parameters

- **folder** (*string*) – path to folder containing jpegs
- **lognm** (*string*) – path to c3p derived csv file

Ciaran Robb, 2019

https://github.com/Ciaran1981/Sfm

This module processes data from the micasense red-edge camera for use in MicMac or indeed other SfM software

This correction is based on the material on the micasense lib git site, though this uses as fork of the micasense lib with some alterations.

mspec.**align_template**(*imAl*, *mx*, *reflFolder*, *ref_ind*, *plots*)

mspec.**mspec_proc**(*precal*, *imgFolder*, *alIm*, *srFolder*, *postcal=None*, *refBnd=1*, *nt=-1*, *mx=100*, *stk=1*, *plots=False*, *panel_ref=None*)
A function processing the micasense data to surface reflectance with a choice of output types dependent on preference.

Notes

A set of RGB & RReNir images is recommended for processing with MicMac

Either 5 band stack or single band outputs can be produced.

Parameters

- **precal** (*string*) – directory containing pre-flight calibration panels pics
- **imgFolder** (*string*) – a directory containing the
- **alIm** (*string*) – 4 digit code of the image to align band images e.g. "0023"
- **srFolder** (*string*) – path to directory for surface reflectance imagery
- **postcal** (*string*) – directory containing pre-flight calibration panels pics
- **refBnd** (*int*) – The band to which all others are aligned
- **nt** (*int*) – No of threads to use
- **mx** (*int*) – Max iterations for alignment (uses opencv motion homography)
- **stk** (*int*) – The various multi-band stacking options 1 = A set of RGB & RReNir images (best for MicMac) 2 = A 5 band stack None = Single band images in separate folders are returned
- **plots** (*bool*) – Whether to display plots of the alignment images for visual inspection
- **panel_ref** (*list*) – The panel ref values unique to your panel/camera If left as None, it will load the authors camera values!

CHAPTER

TWO

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX