Snowpyt

S. Filhol, M. Lanzky

Jun 09, 2023

CONTENTS:

Snowpyt is a Python package to read, vizualize and import snowpit data. Best practice is to import snowpit data in the [CAAML V6 format](http://caaml.org/). An easy way to generate *CAAML* is use [niviz](https://niviz.org/).

Snowpyt allows flexible ploting of snowpyt, including official snow crystal symbols.

CHAPTER

ONE

SNOWPYT: AN OPEN-SOURCE LIBRARY TO VISUALIZE SNOWPITS IN PYTHON

Simon Filhol, November 2016, copyright under the MIT license terms, see the License.txt file

LAST MODIFIED: March 2022 (or see date on github file history)

Feel free to contribute to the project!!!! Many new features can be added...

1.1 To do:

1.1.1 High Priority

- **write class to import and plot CROCUS results**
- **add same colormap for snowgrain type as the snowtool_git for CROCUS**
- write function to save and load pit to and from pickle format (currently not working)
- make ground appear to comfirm the user that the pit reached ground. add note about ground type.

1.1.2 Low priority

- specify the figure size and adjust font size in respect
- render the medatadata text better, convert date to a readable date
- put option to adjust figure size to desired size and dpi. Return axis variable from plotting function for more advanced plotting if needed (i.e. multiple samples)
- add option to save pits in Pickle format or CSV
- add option to save figure in matplotlib format
- add option to plot when multiple sample columns are given.

1.2 Objective

The objective of this library is to provide visualization tool for snowpit data. Started for the need of the Svalbard Snow Research group, this package should evolve to include more snowpit type and visualization scheme.

The snow grain classification follows the guidelines provided by the UNESCO International Classification for Seasonal Snow on the Ground (Fierz et al., 2009)

Fierz, C., Amstrong, R.L., Durand, Y., Etchevers, P., Greene, E., McClung, D.M., Nishimura, K., Satyawali, P.K. and Sokratov, S.A. 2009.The International Classification for Seasonal Snow on the Ground. IHP-VII Technical Documents in Hydrology N°83, IACS Contribution N°1, UNESCO-IHP, Paris.

1.3 Installation

1.3.1 Last stable version from the Pypi repository

Simply run the following in your terminal:

```
pip install snowpyt
```

1.3.2 Last development version for contributing to the project:

Clone the github repository to a local directory using the following command in your terminal

```
git clone https://github.com/ArcticSnow/snowpyt.git
```

or by downloading the package

The branch 'master' consists of the latest stable version. Other develepment versions are included in other git branches.

The package contains all the functions to plot the Snowpyt if library requirements are met. It also contains data samples to test the library. Message us to be added as a contributor, then if you can also modify the code to your own convenience with the following steps:

To work on a development version and keep using the latest change install it with the following

```
pip install -e [path2folder/snowpyt]
```

and to upload latest change to Pypi.org, simply:

1. change the version number in the file `snowpyt/__version__.py`
2. run from a terminal from the snowpyt folder, given your `$HOME/.pyc` is correctly set:

```
python setup.py upload
```

1.3.3 requirements

Python >= 3.6 with the following libraries:

- numpy
- matplotlib
- pandas
- xlrd
- xlm
- skimage
- opencv

1.4 Usage

Currently Snowpyt can be used for two purposes: 1) read and work with CAAML files, and 2) processing NIR images of snowpit to extract refelctance, SSA and optical diameter.

1.4.1 Work with snowpit data: CAAML file

There are three ways to import data into Snowpyt:

1. digitalize your pit with https://niviz.org/ and export your pit as a CAAMLv6 (This format follows an international standard for snowpit). Them use the import_caamlv6() function. More information about the CAAML format
2. input directly data into the snowpit class object

```
from snowpyt import pit_class as pc

################################################################
# Example 1 - using a caamlv6 file:
p = pc.Snowpit()
p.caaml_file= '[PATH TO YOUR FILE].caaml'
p.import_caamlv6()
p.plot(plot_order=['density', 'temperature', 'stratigraphy', 'hardness'])

p.plot(metadata=True)
p.plot(plot_order=['density', 'temperature', 'stratigraphy','crystal size'])

# import isotope values (dD, dO18, d-ex)
p.sample_file = '[PATH TO YOUR FILE].csv'
p.import_sample_csv()

p.plot(plot_order=['dD', 'd18O','d-ex', 'hardness'])
```

The isotope `.csv` file should be following this format:

```
number,height_top,height_bot,dD,d18O,dxs,ice_type
0,94,93.0,-57.55,-8.16,7.73,S
1,93,89.8,-61.56,-8.76,8.54,S
2,89.8,86.6,-75.45,-10.64,9.68,S
```

Many more columns can be added. Create a plotting function to plot any newly named column

1. All the data table are loaded as a Pandas dataframe or Numpy arrays within the snowpyt class object Type the following in your Python console to see the loaded datatable:

```
mypit.table
```

This allows for custom plotting using the library of your choice on top of the existing plotting function

1. Extra Sample Values. Extra column of sample values can be added to the excel file. **Column name must be unique** The current plotting functions will not plot these extra columns, only the first one. However the values are loaded via pandas in the table as a dataframe (see 5.)
2. Compute SWE

```
p = pc.Snowpit()
p.caaml_file= '[PATH TO YOUR FILE].caaml'
p.calc_SWE(method='avg')
```

1.4.2 NIR photography

The `nirpy.py` file contains method to process

```
from snowpyt import nirpy
import numpy as np
import matplotlib.pyplot as plt

fnir = '/home/simonfi/Desktop/202202_finse_livox/NIR_cam/20220224_NIR/DSC01493.JPG'
fcalib = '/home/simonfi/Downloads/Foc0200Diaph028-FlatField.tif'
mo = nirpy.nir(fname_nir=fnir, fname_calib=None, kernel_size=500)

mo.pick_targets()
mo.convert_all()
mo.scale_spatially()
mo.extract_profile(['SSA', 'd_optical', 'reflectance'], param={'method': 'skimage',
                                                               'linewidth': 5,
                                                               'reduce_func': np.median,
                                                               'spline_order': 2})

fig, ax = plt.subplots(1, 3, sharey=True)
ax[0].plot(mo.profile.reflectance, mo.profile.dist)
ax[0].grid(':')
ax[0].set_xlabel('Reflectance [%]')

ax[1].plot(mo.profile.SSA, mo.profile.dist)
ax[1].grid(':')
ax[1].set_xlabel('SSA [mm$^{-1}$]')

ax[2].plot(mo.profile.d_optical, mo.profile.dist)
ax[2].grid(':')
ax[2].set_xlabel('d$_{optical}$ [mm]')
plt.show()
```

1.5 Want to contribute?

Once you have cloned the project to your home directory, create a git branch and here you go. When your edits are stable, merge with the master branch. See this neat tutorial about git branching and merging, here

1.5.1 List of Contributor

- Simon Filhol
- Guillaume Sutter
- Mika Lanzky

1.6 Example

1.6.1 CAAML File Interpretation

Example snowpit

1.6.2 Near-IR Photography

Example snowpit

CHAPTER

TWO

API OVERVIEW

2.1 Modules

- snowpyt
- snowpyt.CAAMLv6_xml: Created on Tue Jul 04 13:32:31 2017
- snowpyt.nirpy: Collection of NIR processing tools
- snowpyt.pit_class: File defining a python class for snowpit data
- snowpyt.snowflake
- snowpyt.snowflake.sf_dict: Created on 6 avr. 2017

2.2 Classes

- nirpy.nir: Class to process NIR snowpit photograph.
- pit_class.Snowpit
- pit_class.density_profile
- pit_class.layer
- pit_class.metadata
- pit_class.sample_profile
- pit_class.temperature_profile

2.3 Functions

- CAAMLv6_xml.childValueNoneTest
- CAAMLv6_xml.get_density: Function to extract density profile from CAAML xml file
- CAAMLv6_xml.get_layers: Function to extract layers from CAAML xml file
- CAAMLv6_xml.get_metadata: Function to extract snowpit metadata profile from CAAML xml file
- CAAMLv6_xml.get_temperature: Function to extract temperature profile from CAAML xml file
- CAAMLv6_xml.has_child
- CAAMLv6_xml.is_node

- nirpy.kernel_square: Function to defin a square kernel of equal value for performing averaging
- nirpy.micmac_radiometric: List of commands to run for deriving a radiometric calibratino profile for the camera
- nirpy.smooth: Function that produce a smoothed version of the 2D array

---

This file was automatically generated via lazydocs.

CHAPTER

THREE

MODULE SNOWPYT.PIT_CLASS

File defining a python class for snowpit data

November 2016, Simon Filhol

3.1 Global Variables

- **snowflake_symbol_dict**
- **path2snowflake**

3.2 class layer

3.2.1 method __init__

```
__init__()
```

3.3 class temperature_profile

3.3.1 method `__init__`

```
__init__()
```

3.4 class `density_profile`

3.4.1 method `__init__`

```
__init__()
```

3.5 class `sample_profile`

3.5.1 method `__init__`

```
__init__()
```

3.6 class `metadata`

3.6.1 method `__init__`

```
__init__()
```

3.7 class Snowpit

3.7.1 method __init__

```
__init__()
```

3.7.2 method calc_SWE

```
calc_SWE(method='avg', ice_layer_density=680)
```

Functino to compute SWE using three methods: avg SWE for all pit 'avg', SWE based on density samples 'samples', and SWE based

Args:

- `method` (str): 'avg', 'samples' or 'layers'. no default - 'avg' is simply the - 'samples' is density by density samples. Top and bottom layer from all top to all bottom to half between density sample 1;2 and N-1;N. All others, make layer horizons between samples, use density sample in middle of these layers as density - 'layers' is density by strat-layer, find matching density sample.Ice layer density a given density.if more than one match, use average. if no matches, search for neareast. (or search for nearest two, upper and lower, and make average)
- `ice_layer_density` (int): assign a constant density to ice layers (An ice layer is detected when hand hardness index = knife = 6)

Returns:

- `float`: SWE in [cm]

3.7.3 method import_caamlv6

```
import_caamlv6(print2term=True)
```

3.7.4 method import_sample_csv

```
import_sample_csv(bar_plot=False)
```

Function to import sample profiles.

Args:

- `bar_plot` (bool): plot sample profile as bar instead of line-scatter. Default is False

---

3.7.5 method `plot`

```
plot(
    save=False,
    metadata=False,
    invert_depth=False,
    figsize=(8, 4),
    dpi=150,
    plot_order=['temperature', 'density', 'crystal size', 'stratigraphy', 'hardness',
↪'sample names', 'dD', 'd18O', 'd-ex']
)
```

Function to plot pit data

Args:

- `save` (bool): save plot
- `metadata` (bool): add metadata to plot
- `invert_depth` (bool): invert depth/height axis. Default is height
- `figsize` (tuple): size of plot in inch. default matplotlib setting
- `dpi` (int): plot resolution, pixel per inches
- `plot_order` (list): list of plots to add to the figure, the order defines the order of the plots

---

3.7.6 method `print_layers`

```
print_layers()
```

3.7.7 method `print_metadata`

```
print_metadata()
```

This file was automatically generated via lazydocs.

CHAPTER

FOUR

MODULE SNOWPYT.NIRPY

Collection of NIR processing tools S. Filhol, March 2022

Inpired by the paper by Matzl and Schneebeli 2016 for more info

A calibration profile was derived from the camera with Micmac. This calibration profile is for correcting vigneting. correct image for vignetting (calib profile is of the size of Raw images. Crop from center to use with jpegs detrend if needed the luminosity, as it can vary linearly from top to bottom of the snowpit sample targets for absolute reflectance calibration (white= 99%, and grey=50%). Fit a linear model Convert reflectance image to SSA with the conversion equation =/

Finally, use the ruler (or other object of know size) in image to scale image dimension to metric system.

TODO:

- write function to extract SSA profile to import in niviz.org
- Raw images are in 12 bit. Find a way to convert to BW from original while maintaining the 12bit resolution. Rawpy might be useful. Then make sure the processing pipeline can accept 12bit data (i.e. all skimage functions)
- wrap micmac function to extract profile 'mm3d vodka'. At least provide the method on how to do it.

4.1 function kernel_square

```
kernel_square(nPix)
```

Function to defin a square kernel of equal value for performing averaging

Args:

- `nPix` (int): size of the kernel in pixel

Returns:

- `array`: kernel matrix

4.2 function smooth

```
smooth(mat, kernel)
```

Function that produce a smoothed version of the 2D array

Args:

- mat: 2D Array to smooth
- kernel: kernel array (output) from the function kernel_square()

Returns:

- 2D array: smoothed array

4.3 function micmac_radiometric

```
micmac_radiometric()
```

List of commands to run for deriving a radiometric calibratino profile for the camera

---

4.4 class nir

Class to process NIR snowpit photograph.

4.4.1 method __init__

```
__init__(
    fname_nir,
    fname_calib,
    highpass=True,
    kernel_size=2000,
    rotate_calib=False
)
```

Class initialization

Args:

- fname_nir (str): path to NIR image
- fname_calib (str): path to radiometric calibration image
- highpass (bool): perform high pass filtering to remove luminosity gradient across the pit
- kernel_size (int): size of the kernel for the highpass filter

4.4.2 method apply_calib

```
apply_calib(crop_calib=False)
```

Function to apply calibration profile to the NIR image.

Args:

- `crop_calib` (bool): if calibration and image are of slightly different size, crop calib and align the two with center.

4.4.3 method convert_all

```
convert_all()
```

Function to convert pixel values to physical values using the targets

4.4.4 method convert_to_SSA

```
convert_to_SSA()
```

Function to convert reflectance to SSA

4.4.5 method convert_to_doptic

```
convert_to_doptic()
```

Function to convert SSA to optical diameter

4.4.6 method convert_to_reflectance

```
convert_to_reflectance()
```

Function to convert image to reflectance using at minimum 2 sets of reflectance targets previously picked

4.4.7 method `extract_profile`

```
extract_profile(
    imgs=['SSA', 'reflectance', 'd_optical'],
    param={'method': <module 'scipy' from '/home/simonfi/miniconda3/envs/dataAna/lib/
↪python3.8/site-packages/scipy/__init__.py'>, 'n_samples': 1000}
)
```

Function to extract profile of values for a list of images

Args:

- `imgs` (list): images from which to sample profile param (dict):
- `method` (str): method to sample the profile. Avail: numpy, scipy, and skimage.
- `n_sample` (int, numpy and scipy method): number of samples along profile
- `linewidth` (int, skimage method): width of the profile
- `reduce_func` (func, skimage method): function to agglomerate the pixels perpendicular to the line
- `spline_order` (int, 0-5, skimage method): order of the spline applied to the sampled profile

examples: {'method': scipy, 'n_samples':1000}, {'method': numpy, 'n_samples':1000}, {'method': skimage, 'linewidth':5, 'reduce_func':np.median, 'spline_order':1}

4.4.8 method `load_calib`

```
load_calib()
```

Function to load radiometrci calibration file

4.4.9 method `load_nir`

```
load_nir()
```

Function to load jpeg NIR images, and convert them to BW

4.4.10 method `pick_targets`

```
pick_targets(reflectances=[99, 50])
```

Function to pick reflectance targets

Args:

- `reflectances` (list of int): List of reflectance targets to pick

4.4.11 method `scale_spatially`

```
scale_spatially()
```

Function to bring real spatial coordinate

Method: 1. click two points 2. provide corresponding length 3. option to provide geometrical correction

---

This file was automatically generated via lazydocs.