pypcd Documentation

Release 0.1.1

Daniel Maturana

Oct 23, 2020

Contents:

CHAPTER 1

pypcd

1.1 What?

Pure Python module to read and write point clouds stored in the PCD file format, used by the Point Cloud Library.

1.2 Why?

You want to mess around with your point cloud data without writing C++ and waiting hours for the template-heavy PCL code to compile.

You tried to get some of the Python bindings for PCL to compile and just gave up.

1.3 How does it work?

1. It parses the PCD header and loads the data (whether in `ascii`, `binary` or `binary_compressed` format) as a Numpy structured array.

2. It creates an instance of the `PointCloud` class, containing the point cloud data as `pc_data`, and some convenience functions for I/O and metadata access. See the comments in `pypcd.py` for some info on the point cloud structure.

1.4 Example

```
import pypcd
# also can read from file handles.
pc = pypcd.PointCloud.from_path('foo.pcd')
# pc.pc_data has the data as a structured array
# pc.fields, pc.count, etc have the metadata

# center the x field
pc.pc_data['x'] -= pc.pc_data['x'].mean()

# save as binary compressed
pc.save_pcd('bar.pcd', compression='binary_compressed')
```

1.5 How to install

You can clone this repo and use setup.py.

```
git clone https://github.com/DanielPollithy/pypcd

cd pypcd

make install
```

1.6 How to run tests

1. Install git lfs: https://github.com/git-lfs/git-lfs/wiki/Tutorial
2. Clone this repo: `git clone https://github.com/DanielPollithy/pypcd` (git lfs will download the assets)
3. `cd pypcd`
4. **Decide whether you want to test all python versions or just your**
    1. All: `make test-all`
    2. Yours: `make test`

1.7 Using with ROS

You can also use this library with ROS `sensor_msgs`, but it is *not* a dependency. You don't need to install this package with catkin – using *pip* should be fine – but if you want to it is possible:

Steps:

```
# you need to do this manually in this case
pip install python-lzf
cd your_workspace/src
git clone https://github.com/dimatura/pypcd
mv setup_ros.py setup.py
```

(continues on next page)

(continued from previous page)

```
catkin build pypcd
source ../devel/setup.bash
```

Then you can do something like this:

```
import pypcd
import rospy
from sensor_msgs.msg import PointCloud2


def cb(msg):
    pc = PointCloud.from_msg(msg)
    pc.save('foo.pcd', compression='binary_compressed')
    # maybe manipulate your pointcloud
    pc.pc_data['x'] *= -1
    outmsg = pc.to_msg()
    # you'll probably need to set the header
    outmsg.header = msg.header
    pub.publish(outmsg)

# ...
sub = rospy.Subscriber('incloud', PointCloud2)
pub = rospy.Publisher('outcloud', PointCloud2, cb)
rospy.init('pypcd_node')
rospy.spin()
```

1.8 Is it beautiful, production-ready code?

No.

1.9 What else can it do?

There's a bunch of functionality accumulated over time, much of it hackish and untested. In no particular order,

- Supports `ascii`, `binary` and `binary_compressed` data. The latter requires the `lzf` module.
- Decode and encode RGB into a single `float32` number. If you don't know what I'm talking about consider yourself lucky.
- Point clouds to pandas dataframes. This in particular is quite useful, since *pandas* is pretty powerful and makes various operations such as merging point clouds or manipulating values easy. Conceptually, data frames are a good match to the point cloud format, since many point clouds in reality have heterogeneous data types - e.g. x, y and z are float fields but *label* is an int.
- Convert to and from ROS PointCloud2 messages. Requires the ROS `sensor_msgs` package with Python bindings installed. This functionality uses code developed by Jon Binney under the BSD license, included as `numpy_pc2.py`.

1.10 What can't it do?

There's no synchronization between the metadata fields in `PointCloud` and the data in `pc_data`. If you change the shape of `pc_data` without updating the metadata fields you'll run into trouble.

I've only used it for unorganized point cloud data (in PCD conventions, `height=1`), not organized data like what you get from RGBD. However, some things may still work.

While padding and fields with count larger than 1 seem to work, this is a somewhat ad-hoc aspect of the PCD format, so be careful. If you want to be safe, you're probably better off using neither – just name each component of your field something like `FIELD_00`, `FIELD_01`, etc.

It also can't run on Python 3, yet, but there's a PR to fix this that might get pulled in the near future.

1.11 It's slow!

Try using `binary` or `binary_compressed`; using ASCII is slow and takes up a lot of space, not to mention possibly inaccurate if you're not careful with how you format your floats.

1.12 I found a bug / I added a feature / I made your code cleaner

Thanks! You can submit a pull request. But honestly, I'm not too good at keeping up with my github :(

1.13 TODO

- Better API for various operations.
- Clean up, get rid of cruft.
- Add a cli for common use cases like file type conversion.
- Better support for structured point clouds, with tests.
- Better testing.
- Better docs. More examples.
- More testing of padding
- Improve handling of multicount fields
- Better support for rgb nonsense
- Export to ply?
- Figure out if it's acceptable to use "pointcloud" as a single word.
- Package data assets in pypi?

1.14 Credits

The code for compressed point cloud data was informed by looking at Matlab PCL.

@wkentaro for some minor changes.

I used cookiecutter to help with the packaging.

The code in `numpy_pc2.py` was developed by Jon Binney under the BSD license for ROS.

1.15 I want to congratulate you / insult you

My email is `dimatura@cmu.edu`.

CHAPTER 2

Installation

2.1 Stable release

To install pypcd, run this command in your terminal:

```
$ pip install pypcd
```

This is the preferred method to install pypcd, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for pypcd can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/dimatura/pypcd
```

Or download the tarball:

```
$ curl  -OL https://github.com/dimatura/pypcd/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 3

Usage

To use pypcd in a project:

```
import pypcd

# also can read from file handles.
pc = pypcd.PointCloud.from_path('foo.pcd')
# pc.pc_data has the data as a structured array
# pc.fields, pc.count, etc have the metadata

# center the x field
pc.pc_data['x'] -= pc.pc_data['x'].mean()

# save as binary compressed
pc.save_pcd('bar.pcd', compression='binary_compressed')
```

CHAPTER 4

History

4.1 0.1.0 (2018-03-15)

- First release on PyPI.

4.2 0.1.1 (2018-03-15)

- Second release on PyPI.

CHAPTER 5

Indices and tables

- genindex
- modindex
- search