Timepix3 analysis toolkit

Installation

To run the converter one can either install the packages manually and run the `tpx3-analyze` file directly or install the package using `pip3`. Assuming you have the (unzipped) folder named `timepix3_analysis` the commands you need are as follows.

Option 1, install packages manually:

```
pip3 install h5py numba numpy matplotlib
./timepix3_analysis/tpx3-analyze --help
```

Option 2, install using pip3. When doing this you need to add `~/.local/bin` to `$PATH`, see this link.

```
pip3 install ./timepix3_analysis
tpx3-analyze --help
```

To uninstall use `pip3 uninstall timepix3_analysis`

The `--help` flag will list out a number of accepted command line options.

The example scripts capable of reading the HDF5 files can be found here:

```
ls timepix3_analysis/examples/*
```

Again `--help` is useful to figure out what they're capable of.

Complete example

Suppose we have collected a number of `.tpx3` files using a detector with four chips. Let us first convert the data to HDF5 and cluster the hits:

```
tpx3-analyze --layout quad *.tpx3 --cluster
```

This generates a file named `out.hdf5`. Any of the example script mentioned before can be used with this file. For example, we can check if there are TDC pulses in the data using:

```
examples/tdc-plot out.hdf
```

Or maybe you're more interested in the TOT image of frame 3:

```
examples/tot-image out.hdf --frame 3
```

Other viewers

To quickly inspect the HDF5 file you can also use generic HDF5 tools. For example, HDFCompass can be

installed using:

```
sudo apt-get installl hdf-compass
```

then `HDFCompass out.hdf` will show you the data that has been saved.

Data format

The HDF5 format used consists of a number of columns, each column simply being a homogeneous array of data.

Column name	Column type	Description
tdc_time	int64	The TDC time in picoseconds
tdc_type	uint8	Either 1,2,3 or 4. Which stands for TDC1 rising, TDC1 falling, TDC2 rising, TDC2 falling respectively
frame_number	uint32	The frame number of the hit (starting from 0)
toa	int64	The arrival time of the hit in picoseconds
tot	uint16	The time over threshold of the hit in nanoseconds
x	uint16	The x-coordinates computed according to the board layout and the `--gaps` parameter
y	uint16	The y-coordinate computed according to the board layout and the `--gaps` paramter, starting from the top of the image. So if `image` is a 2D array the respective pixel value is given by `image[y][x]`.
cluster_index (only with `--cluster`)	uint32	Hits belonging to the same cluster will have the same cluster_index (starts at 0)

The columns frame_number, toa, tot, x, y and cluster_index will have length equal to the number of hits. The columns tdc_time and tdc_type will have length equal to the number of TDC edges registered.

Processing steps performed

As of this writing the `tpx3-analyze` program performs the following processing steps:

- Sorting of the data, both the hits and the TDCs
- TOA rollover correction, in case the hit (or TDC) time data becomes too big
- Board layout rendering, computing the correct x,y coordinates for every hit in the image (also see `--gaps` flag)
- Optionally, clustering is performed. For the cluster algorithm please see the next section

Cluster algorithm

The cluster algorithm works by dividing the sensor area in squares (see `--cluster-square-size`). Every square can hold exactly one cluster index. For every hit the corresponding squares and the neighboring squares are examined for active clusters (first hit is not older than `--toa-diff`). If an active cluster is found the corresponding cluster index is assigned to the hit.

If the central and neighboring squares all contain outdated clusters (older than `--toa-diff`) a new cluster index is produced for the hit in question and assigned to the square containing the hit.

The advantage of this clustering algorithm are:

- It's very fast
- The number of operations per hit has a strict upper bound so it can not get stuck
- It produces cluster indices that are monotonically increasing
- It can recognize a cluster even if it's not well connected (for example because of a dead pixel)

Disadvantages are:

- If two clusters are close in both time and space they might get connected (tweaking `--toa-diff` and `--cluster-square-size` can help)
- The algorithm is only accurate if the hits are sorted in time, so a sorting step is necessary (but usually we need them sorted anyway)