Beamforming Pipeline User Guide

Directory Structure

- *config*: Various configuration files for different pipelines, telescopes
  - *pfb*: PFB filter coefficients
- *matlab*: Transient pipeline simulator written in Matlab
- *plotters*: Some plotters
- *python*: Python helper scripts / prototypes
- *src*: Main source tree
  - *beamformer:* GPU beamforming pipeline (single GPU)
  - *mdsm-multibeam*: Multi-beam, multi-GPU transient pipeline
  - *medicina-beamformer-pipeline*: Beamforming pipeline for BEST-II
  - *medicina-transient-pipeline*: Transient pipeline for BEST-II
  - *prototypes:* Prototype impementations for various algorithms
- *utils*: Some helper scripts

Building the pipeline

1. Go to MDSM directory
2. `mkdir build`
3. `cd build`
4. `cmake ../src`
5. `make`

To install executables do: `sudo make install.` They will be installed in `/usr/local/MDSM`, so add this to the `PATH` environment. Executables generated:

- *standalone-mdsm*: Offline transient detection pipeline, can be used to process filterbank files (or testing)
- *standalone-beamformer*: Offline beamforming pipeline, mainly for testing
- *medicinaBeamformer*: Online beamforming pipeline for BEST-II
- *medicinaTransientPipeline*: Online transient detection pipeline for BEST-II

To run the beamfoming pipeline, simple write: `sudo medicinaBeamformer path/to/config.xml`. The pipeline needs to be run with superuser permission since it allocates memory in kernel space for packet reception. The config file is described below. The output file can be processed/viewed by using the `plot_beams.py` and `process_beams.py` python scripts in `src/beamformer/helpers`

Beamforming Pipeline XML Configuration

The pipeline requires a configuration file to set up the pipeline. Example configuration can be found in the *config* folder. For Medicina use `medicina_debris.xml` as a base configuration. The XML file is split into multiple node, each containing attributes:

- Antennas
  - *number*: Number of antennas in array
- Channels
  - *performChannelisation*: If enabled (1) perform finer channelisation
  - *applyPFB*: If enabled (1) applied an FIR filter prior to channelisation. Filter coefficients can be generated with the *generateCoefficients.py* script in *util*. Generally, generated coefficients are placed in *config/pfb*
  - *ntaps*: Number of taps for PFB channeliser (must match the *NTAPS* macro defined in *src/mdsm-multibeam/params.h* for performance reasons). If this macro is changed the pipeline must be recompiled. A large value will limit the number of samples which can be processed in each iteration, and increase processing time
  - *firPath*: The directory path where PFB coefficients are located
  - *nchans*: Number of subband in raw data stream
  - *subchannels*: Number of fine channels to generate per subband
  - *startChannel*: Start of frequency range (lower subband number) to output to disk
  - *stopChannel*: End of frequency range (upper subband number) to output to disk
- Timing
  - *tsamp*: Sampling time
- Samples
  - *nsamp*: Number of time spectra to process in each iteration. Pipeline will complain if this number is too big
  - *nbits*: bit-length of incoming data
  - *downsample*: Integration factor for writing data to disk. Required when generating multiple beam with no frequency selection. Change this value to match disk I/O limitations
- Writer
  - *filePrefix*: Output file prefix
  - *baseDirectory*: Directory where output files will be written to
  - *singeFileMode*: If enabled (1) all observation output will be generated in one file
  - *secondsPerFile*: If *singleFileMode* is not enabled, observation will be split into multiple files, each of length in seconds as specified in this attribute
  - *usePCTime*
- GPUs
  - *gpuIDs*: GPU Ids (as provided by `nvidia-smi`). Currently only one GPU is supported for the beamforming pipeline

- Beams
    - *antennaFile*: Location of XML file where antennas parameters are defined (generally in `config/antennas.xml`). The total number of beams must match the `BEAMS` macro in `src/mdsm-multibeam/params.h` as well as `BEAMS_PER_TB` if the number of beams is less than or equal to 16, otherwise `BEAMS_PER_TB` must be set a 16.
    - List of beams with the following attributes:
        - *beamID*: Unique beam ID (generally an incrementing integer)
        - *dec*: Pointing declination
        - *ra*: Pointing RA (for tracking beam mode). If 0.0 then beam will be in scanning mode and HA is used to define offset from HA=0
        - *ha:* For beams in scanning mode, define offset from HA=0
        - *topFrequency*: Center frequency of first subband (top of the band)
        - *frequencyOffset*: Subband band width

Setting everything up

TODO...