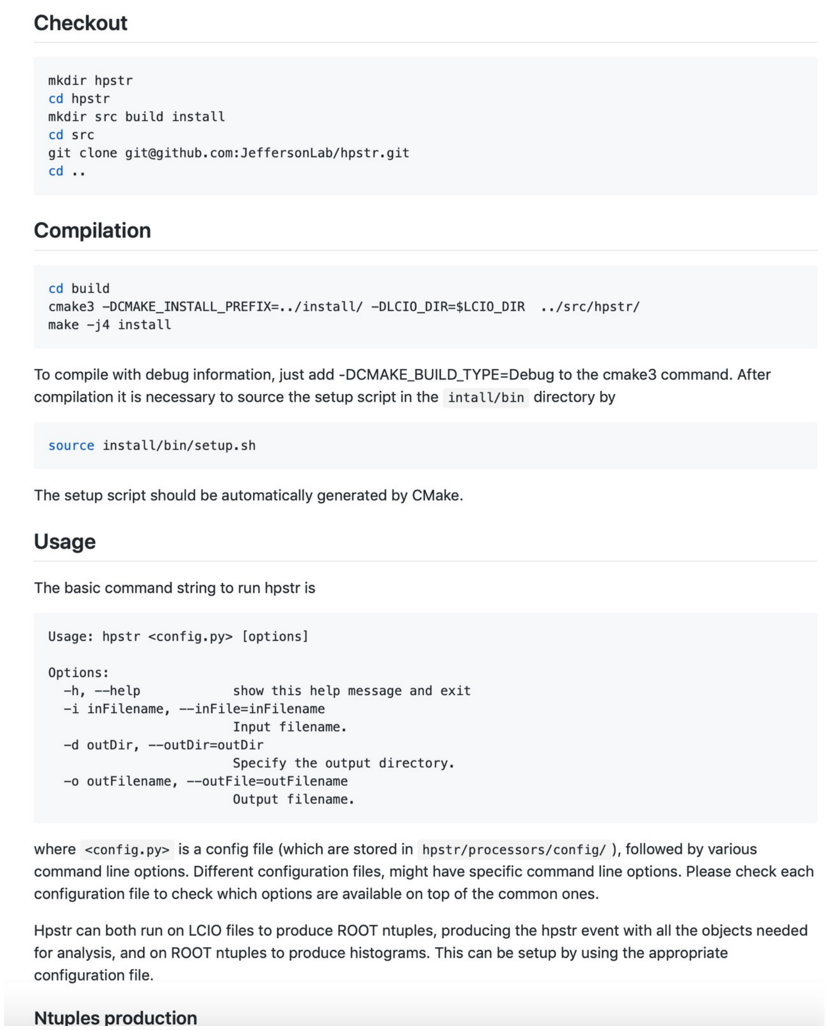
HPSTR updates

PF, Cam

03/11/2020

Introduction

- Hpstr (Heavy Photon Search Toolkit for Reconstruction) is a C++/python package for analysis of reconstructed hps data and MC
- The package supports:
  - Conversion from **LCIO -> ROOT ntuples**, defining a ROOT based EDM with **objects**, i.e. tracks/particles/vertices, and **links** between them
  - ROOT tuples processing to produce **histograms** and/or **flat-Ntuples**
  - Post processing of **histograms** and/or **flat-Ntuples** as well using ROOT python bindings
- The package repository:
  - https://github.com/JeffersonLab/hpstr
  - A README is provided with full instructions from checkout to processing LCIO files



Checkout

```
mkdir hpstr
cd hpstr
mkdir src build install
cd src
git clone git@github.com:JeffersonLab/hpstr.git
cd ..
```

Compilation

```
cd build
cmake3 -DCMAKE_INSTALL_PREFIX=../install/ -DLCIO_DIR=$LCIO_DIR  ../src/hpstr/
make -j4 install
```

To compile with debug information, just add -DCMAKE_BUILD_TYPE=Debug to the cmake3 command. After compilation it is necessary to source the setup script in the `intall/bin` directory by

```
source install/bin/setup.sh
```

The setup script should be automatically generated by CMake.

Usage

The basic command string to run hpstr is

```
Usage: hpstr <config.py> [options]

Options:
  -h, --help            show this help message and exit
  -i inFilename, --inFile=inFilename
                        Input filename.
  -d outDir, --outDir=outDir
                        Specify the output directory.
  -o outFilename, --outFile=outFilename
                        Output filename.
```

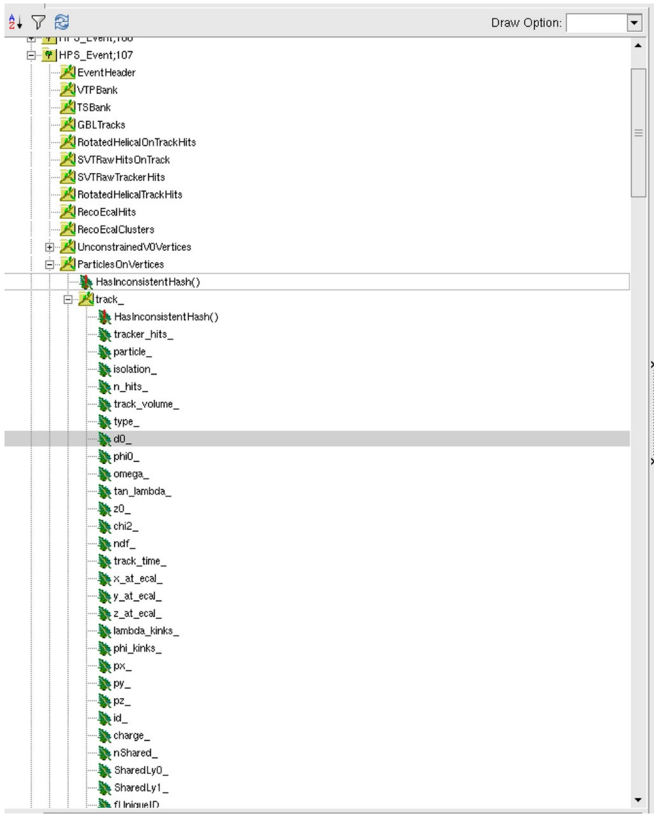
where `<config.py>` is a config file (which are stored in `hpstr/processors/config/`), followed by various command line options. Different configuration files, might have specific command line options. Please check each configuration file to check which options are available on top of the common ones.

Hpstr can both run on LCIO files to produce ROOT ntuples, producing the hpstr event with all the objects needed for analysis, and on ROOT ntuples to produce histograms. This can be setup by using the appropriate configuration file.

Ntuples production

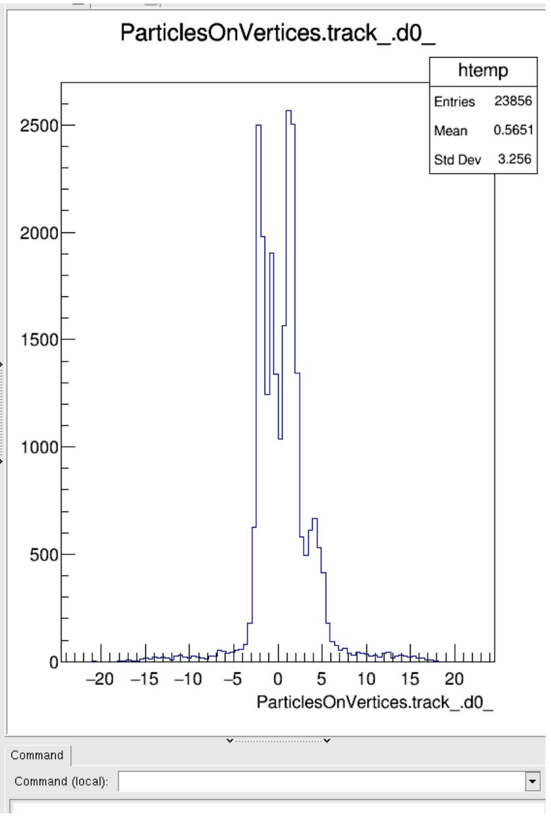
Current Status

- HPSTR now has fully implemented processors for LCIO to a ROOT n-tuple that has the structure for performing both **vtx** and **BH** analysis
- Content:
  - **Unconstrained and TargetConstrained V0s**
  - **Particles** containing associated tracks and clusters
  - **All tracks** in the event containing the **hits on track**
  - **All clusters** in the Calo containing the **Calo hits**
  - **Event Information including trigger bits**



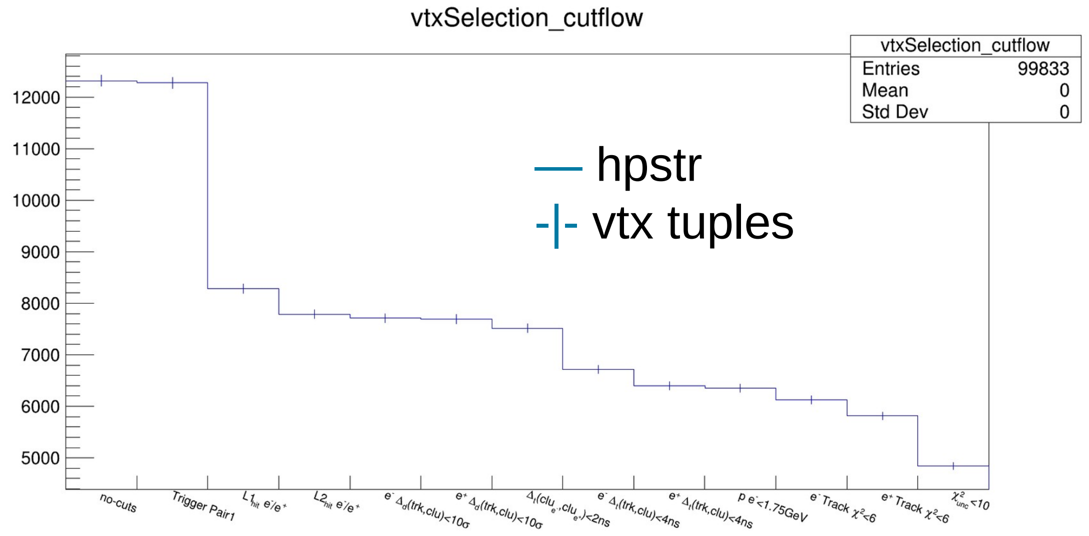
Draw Option:

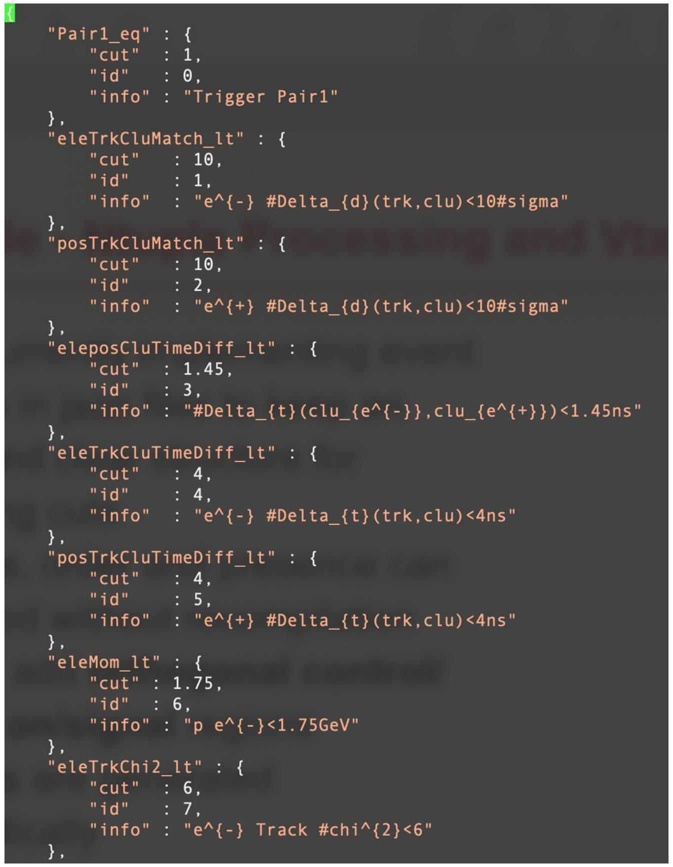
```
HPS_Event;107
    EventHeader
    VTPBank
    TSBank
    GBLTracks
    RotatedHelicalOnTrackHits
    SVTRawHitsOnTrack
    SVTRawTrackerHits
    RotatedHelicalTrackHits
    RecoEcalHits
    RecoEcalClusters
    UnconstrainedV0Vertices
    ParticlesOnVertices
        HasInconsistentHash()
        track_
            HasInconsistentHash()
            tracker_hits_
            particle_
            isolation_
            n_hits_
            track_volume_
            type_
            d0_
            phi0_
            omega_
            tan_lambda_
            z0_
            chi2_
            ndf_
            track_time_
            x_at_ecal_
            y_at_ecal_
            z_at_ecal_
            lambda_kinks_
            phi_kinks_
            px_
            py_
            pz_
            id_
            charge_
            nShared_
            SharedLy0_
            SharedLy1_
            fUniqueID
```



Ntuple Processing and Vtx Cutflow

- Hpstr is currently implementing event selections in json files to keep an ordered and clear structure for bookeeping cuts
- Cut values, order and presence can be changed without recompilation
  - Easy to add **orthogonal control/validation/signal** regions
  - Cutflows are generated automatically
- Vertex preselection cutflow matches between flat-tuple based analysis and hpstr based analysis => **analysis flow validated**

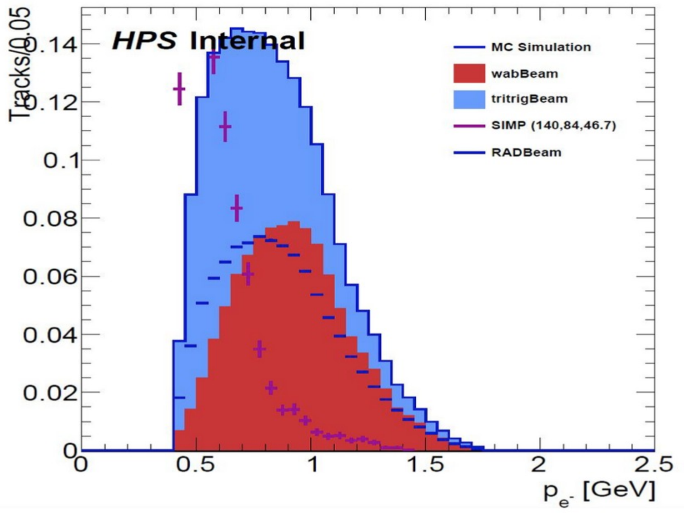



```
{
    "Pair1_eq" : {
        "cut"  : 1,
        "id"   : 0,
        "info" : "Trigger Pair1"
    },
    "eleTrkCluMatch_lt" : {
        "cut"    : 10,
        "id"     : 1,
        "info"   : "e^{-} #Delta_{d}(trk,clu)<10#sigma"
    },
    "posTrkCluMatch_lt" : {
        "cut"    : 10,
        "id"     : 2,
        "info"   : "e^{+} #Delta_{d}(trk,clu)<10#sigma"
    },
    "eleposCluTimeDiff_lt" : {
        "cut"  : 1.45,
        "id"   : 3,
        "info" : "#Delta_{t}(clu_{e^{-}},clu_{e^{+}})<1.45ns"
    },
    "eleTrkCluTimeDiff_lt" : {
        "cut"    : 4,
        "id"     : 4,
        "info"   : "e^{-} #Delta_{t}(trk,clu)<4ns"
    },
    "posTrkCluTimeDiff_lt" : {
        "cut"    : 4,
        "id"     : 5,
        "info"   : "e^{+} #Delta_{t}(trk,clu)<4ns"
    },
    "eleMom_lt" : {
        "cut" : 1.75,
        "id"  : 6,
        "info" : "p e^{-}<1.75GeV"
    },
    "eleTrkChi2_lt" : {
        "cut"  : 6,
        "id"   : 7,
        "info" : "e^{-} Track #chi^{2}<6"
    },
```

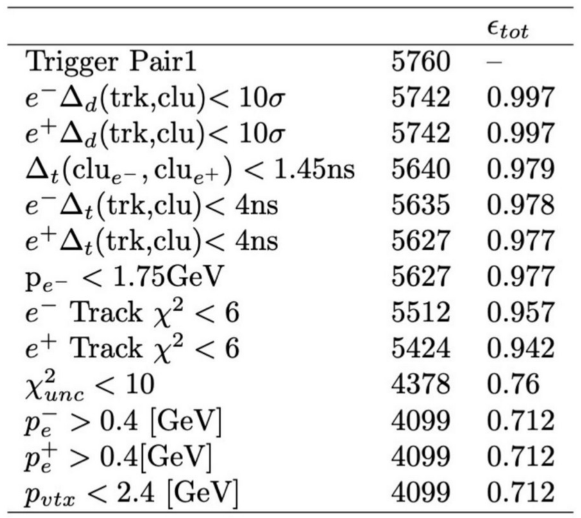
analysis/selections/vtxSelection.json

Streamlined Cutflow Development

- Hpstr provides easy to use support for including different processes:
  - Data / MC
  - A' or SIMPs signal
- Includes scripts for:
  - Shape comparisons and bkg composition
  - Cutflow tables
- Examples here for SIMPs search
- Plots are defined in json files so changes can be made at run time



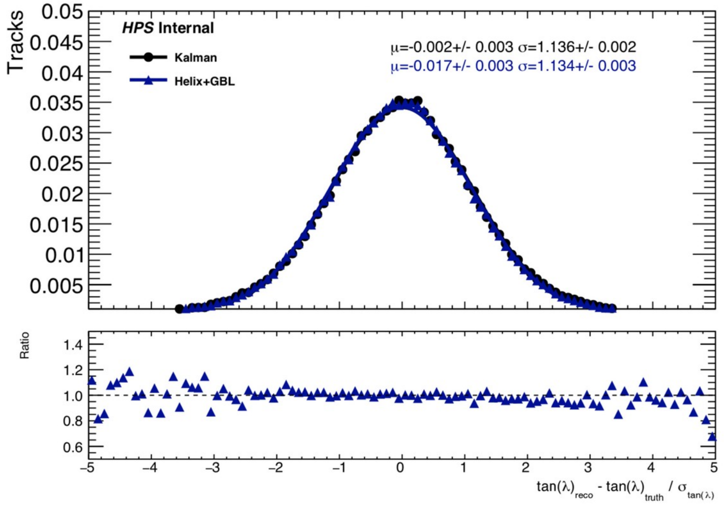
(Stanislava L. S.)

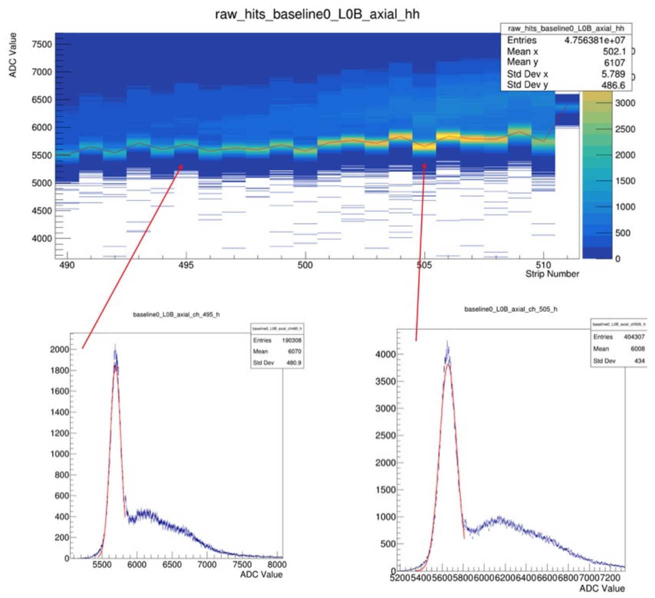


		ϵ_{tot}
Trigger Pair1	5760	–
$e^-\Delta_d$(trk,clu)$< 10\sigma$	5742	0.997
$e^+\Delta_d$(trk,clu)$< 10\sigma$	5742	0.997
$\Delta_t(\mathrm{clu}_{e^-}, \mathrm{clu}_{e^+}) < 1.45$ns	5640	0.979
$e^-\Delta_t$(trk,clu)$<$ 4ns	5635	0.978
$e^+\Delta_t$(trk,clu)$<$ 4ns	5627	0.977
$\mathrm{p}_{e^-} < 1.75$GeV	5627	0.977
e^- Track $\chi^2 < 6$	5512	0.957
e^+ Track $\chi^2 < 6$	5424	0.942
$\chi^2_{unc} < 10$	4378	0.76
$p_e^- > 0.4$ [GeV]	4099	0.712
$p_e^+ > 0.4$[GeV]	4099	0.712
$p_{vtx} < 2.4$ [GeV]	4099	0.712

Tracking / hit studies

- Hpstr provides support for performance studies:
  - **Tracking analysis**, i.e. Kalman-GBL comparison, track efficiency, truth matching…
  - Baseline extraction for **SVT hits** including gaussian+landau fit for charge deposition and baseline extraction
  - **Calo/Hodo hits/cluster studies** in principle would be simple to add
- Analysis flow, radiative fraction and statistical interpretation for BH analysis are included in the framework





Current HPSTR Status

- Hpstr framework is **in good shape** to support 2019 analysis
  - **Vertexing analysis flow validated up to preselection**
    - Analysis flow in course of validation but:
      - **Radiative Fraction** can be extracted (and agrees with Rafo’s)
      - **Statistical interpretations** ported from old repos to hpstr
- **New users** (Stany) whom joined HPS recently were able to produce first results quickly on SIMPs searches
  - **README with working examples** and default processing documented
- Hpstr also useful to do **performance studies** relative
  - Tracking, hit/cluster studies in SVT/Calo/Hodo …
- **Plotting** support with an “hps” style available
- Dev on C++ portion becoming less of a priority

HPSTR in the Future

- Focus shifting towards development of python analysis tools
    - This portion runs on the output of C++ steps
    - Currently organization of python code is poor
    - Need to put effort into having a properly organized python project
- The python framework is more than JUST plotting
    - Divide histograms for acceptance and eff studies
    - Make new histograms from unbinnded data in selection output
    - Integrate vtx efficiency and calculate expected A’ rate
    - Most of the analysis level stuff can happen in this layer
- Further development of C++ portion can probably be handled by the handful of people who wrote most of it to begin with