Cluster Finding

#hackaton 11 Jan 2019

J. Monroe (RHUL)

Strategies

1) Gaussian blur to find contiguous pixels above threshold
   (i) blur pixels with gaussian kernel of (physics-driven) size
   (ii) iterate (i) with tight cuts to determine rms to define threshold Q
   (iii) associate neighboring ‘superpixels' with Q > threshold
   (iv) apply smart strategies (Radon transform, Hough transform, etc.)

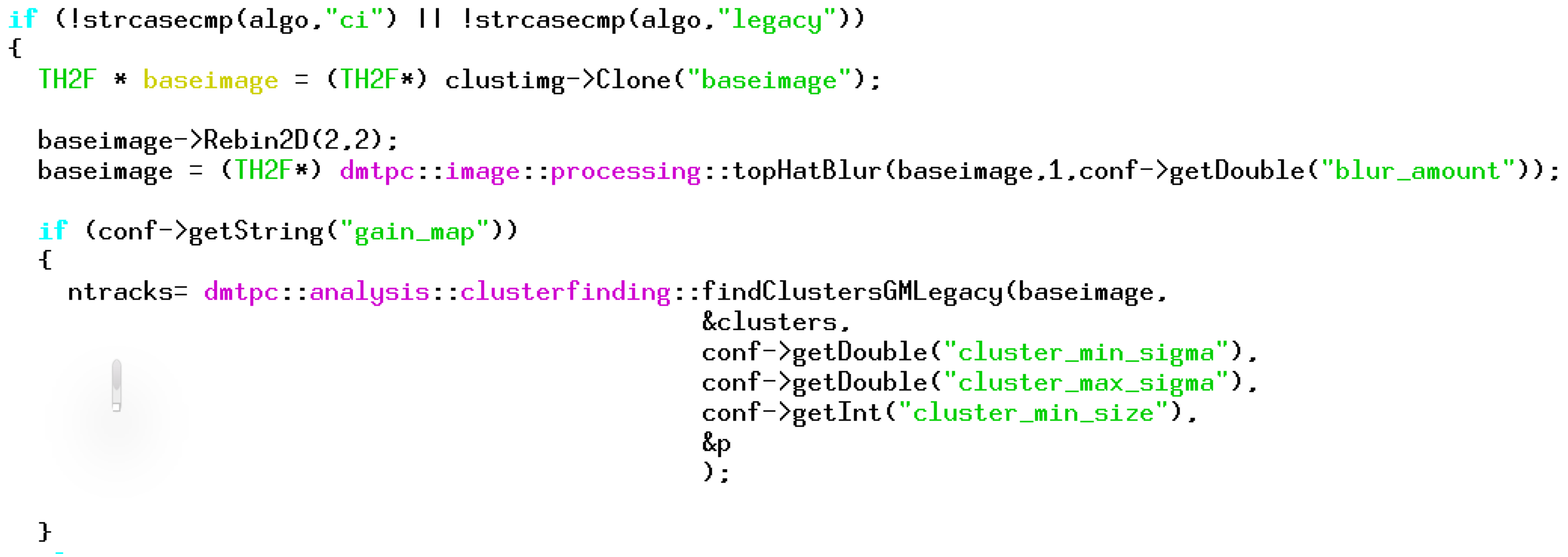
2) Time series analysis
   (i) calculate per pixel pedestal with loose cuts
   (ii) iterate (i) with tight cuts, determine rms dQ/dt to define threshold dQ/dt
   (iii) search for dQ/dt in each pixel
   (iv) associate neighboring pixels with dQ/dt > threshold

Existing Code

1) DMTPC implementation: see section 4.2 of Deaconu thesis

code is here: /home/dmatter/Software/hptpc-daq
DmtpcAnalysis/src/cleanSkimFunctions.cc
dmtpc::analysis::cleanskim::Functions::clusterFind
has various algorithms available:

(i) legacy



```
if (!strcasecmp(algo,"ci") || !strcasecmp(algo,"legacy"))
{
  TH2F * baseimage = (TH2F*) clustimg->Clone("baseimage");

  baseimage->Rebin2D(2,2);
  baseimage = (TH2F*) dmtpc::image::processing::topHatBlur(baseimage,1,conf->getDouble("blur_amount"));

  if (conf->getString("gain_map"))
  {
    ntracks= dmtpc::analysis::clusterfinding::findClustersGMLegacy(baseimage,
                                                                    &clusters,
                                                                    conf->getDouble("cluster_min_sigma"),
                                                                    conf->getDouble("cluster_max_sigma"),
                                                                    conf->getInt("cluster_min_size"),
                                                                    &p
                                                                    );


  }
```

Existing Code

(ii) seeded cluster finding

-iterative reduction of threshold value, calculates probability of cluster objects

```
if (!strcasecmp(algo,"seed"))
{

  if (!stitch)
  {
    ntracks = dmtpc::analysis::clusterfinding::findClustersGMSeed(clustimg,
                                                   &clusters,
                                                   conf->getDouble("cluster_find_seed_seed_threshold"),
                                                   conf->getDouble("cluster_find_seed_thresh_ring_ratio"),
                                                   1,
                                                   conf->getDouble("cluster_find_seed_min_threshold"),
                                                   conf->getInt("cluster_find_seed_blur_radius"),
                                                   conf->getDouble("gaussian_blur_amount"),
                                                   conf->getInt("cluster_neighbors_threshold_for_filling")
                                                   conf->getInt("cluster_min_neighbors_to_keep_pixel"),
                                                   conf->getInt("cluster_min_size_unbinned"),
                                                   &p,
                                                   conf->getBool("seed_cluster_find_reproduce_v4_bug")
                                                   );


  }
```

Existing Code

(iii) anisotropic diffusion

-varies the blur kernel size depending on the image 2D gradient

```
else if (!strcasecmp(algo,"ad"))
{
    ntracks = dmtpc::analysis::clusterfinding::findClustersADHysteresisGM(clustimg,
                                                           &clusters,
                                                           conf->getDouble("cluster_find_ad_k"),
                                                           conf->getDouble("cluster_find_ad_lambda"),
                                                           dmtpc::image::processing::TUKEY,
                                                           conf->getInt("cluster_find_ad_niter"),
                                                           conf->getDouble("cluster_find_ad_gradient_blur"),
                                                           dmtpc::image::processing::SOBEL,
                                                           conf->getDouble("cluster_find_ad_high_thresh"),
                                                           conf->getDouble("cluster_find_ad_low_thresh"),
                                                           conf->getInt("cluster_neighbors_threshold_for_fill
g"),

                                                           conf->getInt("cluster_min_size_unbinned"),
                                                           &p
                                                           );
}
```

Existing Code

(iv) bilateral filter

-uses distance between pixels above threshold and difference between pixel values

```
 else if (!strcasecmp(algo,"ring"))
 {
   char * debug = 0;
def DEBUGOUT
   char buf[64];
   debug = buf;
   sprintf(debug,"debug/ring%d.root",nth);
dif
   ntracks = dmtpc::analysis::clusterfinding::findClustersGMRing(clustimg, &clusters,
                                                    &reduced_clusters, stitch,
                                                    image_rms,image_mean,
                                                    conf->getDouble("cluster_find_ring_space_sigma"),
                                                    conf->getDouble("cluster_find_ring_rms_sigma"),
                                                    conf->getDouble("cluster_find_ring_core_thresh_high"),
                                                    conf->getDouble("cluster_find_ring_core_thresh_low"),
                                                    conf->getDouble("cluster_find_ring_ring_thresh"),
                                                    conf->getDouble("cluster_find_ring_ring_nsigma"),
                                                    conf->getInt("cluster_min_size_unbinned"),
                                                    dmtpc::image::processing::BILATERAL_GAUSSIAN,
                                                    conf->getInt("cluster_find_ring_ncleanup"),
                                                    &p,
                                                    debug, 12

       );
```

Yet More Code

(ii) <u>ClusterFinding.cc</u>
 -functions for cluster finding algorithms

(iii) <u>ClusterAlgo.cc</u>
 -functions to estimate cluster properties

2) Time series analysis
 (i) rudimentary per-pixel threshold calculation exists in (I think(?))
 /home/dmatter/Software/hptpc-daq/DmtpcSkim/src/CcdCalibMaker.cc

 Nothing else implemented in DMTPC for the time-series strategy.
 See JPL CCD data analysis, post-exposure image sharpening

Comments

We are in the situation where the hard part is identifying the interesting pixels—we will likely need strategies where we can identify pixels with <3 sigma excursions as “interesting”.

(i) HPTPC should implement “legacy” and “seed”, without stitching, in raptorr. These reply on root functions so should be straightforward.

(ii) time series is promising for us…