ACD Pattern Recognition

While trying to measure the efficiency of the ACD and study the effects of the gaps I came up against a problem that Robert Johnson had already mentioned.
Namely we don't keep track of how often a track just missed a tile as we didn't see any signal in the tile.

Put another way, the information we calculate for the ACD come from one of two incomplete sources,

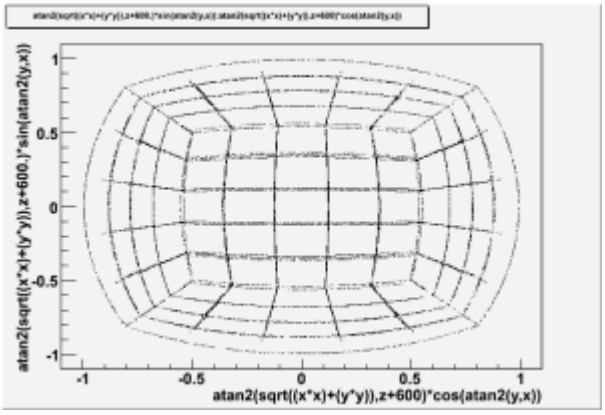
1. We calculate distances of closest approach/ active distances for combinations of tracks with "hit" tiles and ribbons, where hit means "above zero suppresion". In the Recon file these are stored as AcdTkrHitPoca objects. These are used to fill all the merit variables.
2. We use the GEANT model to get all the intersections between the tracks and the ACD using the G4Propagtor. This is nice, but suffers the major (critical?) flaw that it is binary. Either we hit a particular sensor or we didn't, GEANT doesn't have any tolerances. The data are store in the recon file as AcdTkrIntersections and are only used for calibrations & detector studies.

In the last couple days I played around with a better way of handling the question, "which tiles & ribbons did the track come close to?" The idea is to use a two step process:

1. Do something clever to get the set of tiles and ribbons the track came close to,
2. For each selected tile/ribbon do all the same calculations as usual.

most of the rest of this post is about 1, the "something clever".
The idea is to map the ACD into 2 dimesions and then use a lookup table to build up the set of possible relevent tiles & ribbons.

This figure shows one such mapping. It is the polar coordinate of the ACD, as seen from a point 600mm below the Z=0 plane of the LAT.



Given x,y,z the transformations to the new coordinates (u,v) are:

```
double dist2 = x*x + y*y;
double dist = sqrt(dist2);
double zVal = z + 600.;
double rad = atan2( dist , zVal );
double u = rad * ( x / dist );
double v = rad * ( y / dist );
```

Then we divide the u,v plane into 20 segments in each direction, making 400 possible bins.

At the start of the reconstruction job we pre-compute the look-up table. To build in tolerance we increase the size of all the detector elements by some amount (say 200mm), then we scan across the detector element in 10x10 steps (100 points per detector) for each point we get the (u,v) bin the point falls in a add this to the overall map. Technically speaking the map is

```
std::map< AcdRecon::AcdUVBin, std::list< idents::AcdId > >
```

Then for each track in each event we

1. Calculate where the track crosses a cube located at the nominal position of the ACD
2. Extract the associated (u,v) bin
3. Look up the list of AcdId's for the detector elements that come within the tolerance (say 200mm) of that (u,v) bin

4. Perform the usual Acd distance of closest approach/ active distance calcualtions for each of those detector element

The code for all of the above is written as of 10 sept, 2008. Then main remaing issues are:

1. How to get the new information into the recon/ svac/ merit files
2. How to do 1) without changing any existing information

I propose to procced as follows:

1. Continue to calculate and fill the AcdTkrHitPoca for all combinations of tracks and ACD tiles/ribbons with signals
2. Store the new information with the existing AcdTkrHitPoca structures in the recon file. Those structures are designed to store all the distance information.
3. This means that the list of AcdTkrHitPocas will include both cases where the relevent detector has a signal and those where it does not.
4. Add new data fields to the AcdTkrHitPoca structure to store the MIP equivalent signal seen in the ACD detector element. This will allow us to distiguish quickly between tiles near the track that have signal and those that don't.
5. Update AnalysisNtuple to ignore those AcdTkrHitPoca's that have no signal when filling merit variables, this will insure that the merit variables are identical to what we have now.
6. Improve the logic in AnalysisNtuple to fill new merit variables that take into account signal size, distance of closest approach and track quality to find the tile/track combination most likely to VETO the event

   Here's a copy of the ACD Tile Map for posterity...