Summary of Various EEMC Reconstruction Codes
Draft v1.2

S. Gliske

June 6, 2011

1 Introduction

This document is organized such that each section covers a specific task, with subsections for each known approach. For each approach, the original authors and implementers are listed (when known) and a brief summary is provided. The collaborators who have written most of the code include[1] Jan Balewski, Alice Bridgeman, Pibero Djawotho, Weihong He and Jason Webb. In addition to EEMC code, some methods used for the barrel EMC (BEMC) are also given for comparison. Many of these methods for the BEMC are either written or used by Mike Betancourt for the inclusive photon analysis. These are often associated with the `StGammaMaker` class. A summary of code from 2004 is available in Ref. [2], along with descriptions of some major trunks of the CVS repository relevant to the EEMC. The γ+jet analysis of Seema Dhamija, using the end cap, follows the "skeleton" of Alice Bridgeman and also uses the `StGammaMaker` class.

Although not available for the current version of this document, it would also be useful to collect a summary of methods used by other collaborations for similar detector components. As many of us have colleagues working with LHC or Fermi Lab experiments, a brief synopsis should be not to difficult to obtain.

2 Clustering Algorithms for Towers, Pre-Shower and Post-Shower

Clustering is generally not preformed on the pre-shower and post-shower layers. Two common alternatives exist. Often, the information from the pre-shower and post-shower layers are taken just for the segment directly covering the SMD point. In other cases, the segments of the pre-shower and post-shower are put into clusters to match the clustering results of the towers. However, when clustering is applied directly to the pre-shower and post-shower layers, the algorithms are set to be exactly used for the towers.

2.1 3×3 Method

This method is used for the BEMC Towers (BTOW), and is hence used by Mike Betancourt. The code for this method is in the `StBarrelEmcClusterMaker` class [3]. Clusters are defined as the 3×3 block of towers surrounding a tower above a certain threshold (a seed tower). If two (or more) clusters are not sufficiently separated, then both (all) overlapping clusters are removed. In this context, sufficiently separated means that the center (i.e. seed) tower of one cluster is not contained within another cluster. Non-seed towers are allowed to belong to multiple clusters, and no correction is made for sharing the energy of such towers that belong to multiple clusters.

2.2 MineSweeper

This method was developed by Jan Balewski [4].[2] The code is implemented in both the `StEEmcIUClusterMaker` class [5] (by Weihong He[3]) and the `StEEmcClusterMaker` class [6] (written by Jason Webb). There seems to be no reason behind the multiple copies of the code in different classes.

[1] For a more detailed, though dated, list of people involved, see Ref. [1] from 2004.
[2] Note that Refs. [5] and [6] have the incorrect url for this page. The correct url is Ref. [4] of this document.
[3] The thesis [7] does not indicate which clustering algorithm is used for the towers.

This method is very similar to the method used for clustering the BTOW (Section 2.1), with two important differences. First, for clusters where the seeds are within the 3×3 adjacency block of another cluster, `MineSweeper` keeps the cluster for which the seed has highest energy. The method of Section 2.1 keeps none of the clusters in this case. Also, towers which are members of multiple clusters are given a weighted membership in each cluster, with the weights being proportional to the energy of the seeds of the relevant clusters. The weights for each given tower are normalized to one.

2.3 Continuous Blocks

This method also "grows" clusters from "seed" clusters. The method has been coded by Jason Webb and is also used by Alice Bridgeman. The code is located in both the `StMyClusterMaker` class [8], within the `StEEmcPool`, and the `StAbClusterMaker` class [9]. Again, there is no reason for the duplication of identical code.

This method can be viewed as a simplification of the DBScan algorithm (See Section 3.1). Clusters are defined as sets of towers, such that the towers of each set form a continuous block and that all towers (in all sets) have energy above a given threshold. These sets (clusters) are determined as follows [8]:

> `The tower cluster finder starts by identifying all towers (or pre/post elements) which exceed a user-specified seed threshold. The list of seed towers is then sorted by energy. We loop over all seed towers starting with the most energetic and proceeding to the least energetic.`
>
> `For each seed tower, we create a new cluster. We add all neighboring towers which exceed a specified minimum energy. These towers are flagged and will not be added to any other clusters. We then treat all towers within the cluster as a seed tower, and add any neighboring towers above threshold to the cluster. This procedure repeats until no new towers can be added to the cluster, or a user-specified size limit (dEta,dPhi) is reached.`
>
> `The cluster is then added to the list of clusters, and the next seed tower (not previously flagged) is processed.`

3 Clustering Algorithms for the SMD

Note, some π^0 studies have been carried out without use of the SMD [10].

3.1 DBScan

DBScan is used as a "pre-clustering" method by Alice Bridgeman, as it does well at distinguishing noise from background, but does not do well at distinguishing overlapping clusters. DBScan is implemented in the `StAbClusterMaker` class [9]. The method proceeds much like the "seed" methods for tower clusters, especially similar to the method in Section 2.3. DBScan includes several important refines to account for noise in the data. Further details can be found in Ref. [11]. DBScan is a very commonly used algorithm in many fields [11].

3.2 *k*-means Clustering

Alice Bridgeman applies the k-means algorithm [12, 13] to "meta-clusters" outputted by DBScan, with k fixed at 2. She then had some test to determine whether the 2-way split was preferable over not splitting the meta-cluster. Alice Bridgeman also considered a fuzzy c-means algorithm [14], but commented that it often resulted in multiple clusters with equivalent means and widths [13]. Further details regarding the choice of initial clusters is located in Ref. [13]. Note: c-means and K-means are closely related to mixture models for a specific kernel function.

3.3 A Seed-Based SMD Cluster Method

This method is implemented in `StMyClusterMaker` [8], which was written by Jason Webb. The general approach is similar to the seed-based clustering algorithms for the towers. In the code for this class is the following text:

> `The SMD cluster finder starts by identifying all SMD strips in a given plane which are above a user-specified seed threshold. The list of seed strips is then sorted descending in energy. We loop over all seeds starting with the most energetic and proceeding to the least energetic.`
>
> `For each seed strip, we create a new cluster and add the strip to it. Neighboring strips are added until one of the following conditions are met:`
>
> 1. `A strip falls below a user-defined minimum`
> 2. `The next SMD strip exceeds the energy of the current strip`
> 3. `The cluster fails to grow by a user-specified amount`
> 4. `We have reached a user-specified size limit`
>
> `Each strip added to a cluster is flagged and will not be used by any other clusters. The SMD cluster is added to the list of clusters if it has more than a user-specified number of strips, and the next seed strip (not previously flagged) is processed.`

3.4 Another Seed-Based SMD Cluster Method

This method is implemented in `StEEmcIUClusterMaker` [5], written by Weihong He. This method is fairly similar to that in Section 3.3. One of the most notable differences is that an envelope is placed around each seed, increasing the seed threshold for neighboring strips.

Quoting from [5]:

> `The SMD clustering algorithm starts by finding all strips which exceed a user-specified seed threshold. These strips are sorted by energy, and we work from the most energetic seed to the least energetic seed.`
>
> `Strips adjacent to the SMD seed are added to the cluster and removed from the pool of seed strips. The number of adjacent strips added is specified by the user in the setMaxExtent(max) method and the default strip number is set to 3 to both sides around a seed strip.`
>
> `Once a cluster is identified, the seed threshold is raised in the vicinity of that cluster, including a special floor setting described below. For details, see`
> `buildSmdClusters().`
> `...`
> `In order to form a cluster, we require a minimum of three SMD strips. Fewer strips suggests a MIP, although low-energy EM showers (<∼1 GeV) can deposit energy in only two consecutive SMD strips.`

More details can be found in Ref. [7]. The effects of varying the available parameters have been studied by Keith Kruger [15]. In particular, for low p_T events, setting `setMaxExtent` to 3 causes a strange bump below the π^0 in the $M_{\gamma\gamma}$ plot. This bump is removed by setting `setMaxExtent` to 6. However, for high p_T events, setting `setMaxExtent` to 6 causes an asymmetric π^0 peak and greatly reduces the statistics. Keith also considered a method which starts with `setMaxExtent` set to 3 and increases the cluster with the next adjacent strip, providing the adjacent strip has non-zero energy deposition and less energy deposited than the previous strip. This method also failed on the high p_T events, yielding an asymmetric peak. Additionally, Keith's experience suggested that the seed floor requirement is not productive, as there are single spikes in the SMD strips. Placing the seed floor around the single spikes decreases the chance for the algorithm to see the legitimate peaks nearby the spike.

3.5 And Yet Another Seed-Based SMD Cluster Method

There exists a Drupal blog [16] describing implementing the method used in the barrel (See Section 3.6) for the end cap SMD. The method is implemented in the `StGammaPointMaker` class [17], although the code has been marked as "no longer relevant" and removed from the CVS repository.

3.6 Methods Used with the Barrel EMC

The relevant "maker" for clustering the BSMD strips is the `StPreEclMaker` [18]. Only a self proclaimed "old" method is available in the CVS repository, as of date. The method is contained in the `StEmcOldFinder` class [19], which is an implementation by Alexandre A. P. Suaide of algorithms of Subhasis Chattopadhyay and Aleksei Pavlinov. There is also a Drupal blog regarding the method [20]. The method starts with the largest hit in a "module" (what they call the divisions of the towers, pre-shower and BSMD) as the seed, which a minimum energy threshold for a strip/tower to be a seed. Adjacent towers/strips are added if they are above some threshold value. There is also a cut on the maximum number of hits in a cluster and the minimum energy of the cluster. Note that adjacency is defined by the index being numerically adjacent, not using a "getNeighbors" like function.

For Mike Betancourt's photon+jet analysis, the BSMD is not used for a fine point resolution, but is instead only used for discriminating between whether one or two clusters are present. It was decided that a a finer point resolution is not needed. Thus, a SMD strip clustering algorithm is not utilized. Instead, a Bayesian method for comparing a single Cauchy[4] versus a double Cauchy model is used to discriminate between towers with one or two hits. The main portion of this code seems to be in the `StGammaFitter` class [21], written by Pibero Djawotho, and the `StGammaCandidateMaker` class [22], written/modified by Mike Betancourt, Pibero Djawotho, and Jason Webb.

4 Point Makers

The historical note should be made, that the point maker was corrected in January, 2007, to account for the difference in the z-positions of the two SMD layers and the direction of the incident particle. Ignoring the difference in z led to parallax errors. This fix was implemented in v1.5 of `EEmcSmdGeom` [23].

4.1 Weihong's Point Maker Method

Quoting from Ref. [24]:

> `This class produces points using SMD clusters found from an instance of StEEmcIUClusterMaker.`
>
> `A point is defined as the coincidence of a cluster in each SMD plane with an active tower.`
>
> `Points are found in the following order:`
>
> 1. `Loop over all SMD clusters beginning with those closest to the beam and working radially outwards.`
> 2. `Any unique U,V pair below an active tower is considered to be a point regardless of the energy match between the pair. When we find such a match we remove the U and V clusters from the pool of SMD clusters and search for another match. (goto 1).`
> 3. `If no unique pair was found, we select the U,V pair with the closest match in energy between the two planes, if the pair is underneath an active tower, it is identified as a point candidate. Once we find all point candidate n the sector, we check to see if we need to apply splitting algo to save more points according to energy matching. Then, we remove the pair of clusters from the pool of clusters, and repeat. (goto 1).`

[4] The Breit-Wigner mass distribution is a Cauchy distribution.

4. The algo proceeds to the next sector.
5. The energy of a point is decided by tower cluster energy. The position of a point is decided by SMD information.

The details of computing the energy of the point is specifically [24]:

```
/// Point energies were initially set to be the energy
/// deposited in the SMD. We will now overwrite these
/// energies, calculating the equivalent EM energy using
/// a sampling fraction of 0.7
```

The source of the 0.7% sampling fraction is unclear, though it shows up in many of the codes. The code following the above comment divides by 0.007 and again by 2. There is no comment in the code or documentation regarding the extra factor of two. It is presumed to be due to the fact that every particle passes through two SMD strips per layer, as the strips have an overlapping triangular structure. As with the SMD clustering algorithm, the effects of varying the available parameters have been studied by Keith Kruger [15]. There is also a cut placed on the asymmetry of the cluster energies between the two SMD layers, though studies by Keith Kruger have shown that the this cut is no longer relevant [15].[5]

4.2 Jason's Point Maker Method

Quoting from Ref. [26]:

> The point making algorithm works as follows. Clusters are input from the user's algorithm (see StEEmcGenericClusterMaker). Tower clusters are ranked, descending in energy. For each tower cluster, we look at the SMD clusters which were matched to it by the cluster maker and find the best matches between the u, v planes by minimizing the following sum

$$\chi^2 = \sum_i^{\min(N_u,N_v)} \frac{E_u^i - E_v^i}{N_{\text{mips}}^i}. \tag{1}$$

> After all points have been found, we attempt to split points in the case where 2u + 1v or 1u + 2v clusters were found matched to a tower cluster. When this is the case, we will split the singleton cluster only if it improves the χ^2 equation above.
>
> The algorithm which is used to split the cluster works as follows. The two clusters in the resolved view (R1 and R2) act as the fitting functions in the merged view. The seed strip of R1 is aligned with the seed strip of the cluster in the merged view (M). The seed strip of R2 is aligned with the leftmost strip of M, and we compute a strip-by-strip χ^2. The seed strip of R2 is scanned across M and a χ^2 determined for each position. We then swap R1 for R2 and repeat. The position and ordering which minimizes χ^2 is then used to split M.
>
> In the case where an SMD cluster has been split, the resulting pair of clusters will have the same "unique" ID as the parent cluster.

In the above quote, the value N_{mips}^i is not defined. From looking at the code, it appears to be defined as $(E_u^i + E_v^i)/1.3$, with no comments how the magic number 1.3 is derived. Note also, the code actually uses the formula

$$\chi^2 = \sum_i^{\min(N_u,N_v)} \frac{(E_u^i - E_v^i)^2}{N_{\text{mips}}^i}, \tag{2}$$

whereas the comment is missing the square on difference in the numerator. The interpretation of this value as a χ^2 is unclear.

[5] Weihong's studies regarding this cut were based on faulty simulations that had all air replaced by lead in the EEMC. [25]

4.3 Alice's Method

Alice claims to mostly follow Jason's method: "The big difference is that I no longer loop by tower clusters" [13]. Alice also comments:

> "I have found that splitting clusters (when you have more clusters in one plane than another) seems to cause funny looking *[digamma]* mass distributions. However, even without splitting, the algo appears to be more efficient than the IUCF one." [13]

She explains this alternately as

> "I do not attempt splitting if more clusters are found in one plane than the other. With my clustering algorithm this seems to result in very funny looking clusters." [27]

4.4 Barrel Point Maker Method

The method is located in `StPointCollection` [28], with the original author and date of Subhasis Chattopadhyay and January 2000. This class is part of the `StEpcMaker` [29]. See also Ref. [30]. Points in the two BSMD are associated such that the magnitude of the asymmetry in the energies

$$\left| \frac{E_\eta - E_\phi}{E_\eta + E_\phi} \right| \tag{3}$$

is minimized, where E_η, E_ϕ are the energies from the η and ϕ BSMD layers. Note, Mike Betancourt (i.e. `StGammaMaker`) includes no point maker utilizing the SMD strips. See Section 3.6.

5 Determining the number of hits

In all cases, it seems the number of hits is simply the number of valid points given by the point maker algorithm, with only one exception.

5.1 CDF Conversion Method

A conversion method, based on methods used at CDF, was studied by Jason Webb [31]. Note, the method is for correcting yields (counts), and not for individual events. The method determines the gamma yield given the total yield (including background). It is possible that the ideas considered could yield some information on the event level, though more work would need to be done to accomplish this.

6 Tower Energy Sharing Scheme

6.1 Alice

This method shares the tower energy as follows. Energy sharing is used when a tower includes more than one SMD "point" in the 3×3 block around the tower. The tower energy contributes to a given SMD point with weight equal to the SMD energy of the point divided by the total energy of all SMD points in the 3×3 block. The pre- and post-showers values for the points are, however, just set to the value of the immediate block the point lies within, with no sharing and no summing over neighbors. Full details can be seen in the latter portion of the `StAbPointMaker::Make()` member function [27].

6.2 Jason/Weihong

The class `StEEmcIUPointMaker` [24] has three implementations for energy sharing, labeled `shareEnergy`, `shareEnergySimple`, `shareEnergySMD`. Only `shareEnergySimple` is currently enabled. The `shareEnergySMD` method is effectively the same method as described in Section 6.1. The `shareEnergy` method contains the following comment

```
/// Perform energy sharing between points.  We fit the tower
/// response (analytically minimize chi^2 for the identified
/// smd points.
```

However, the method can yield negative energies, as it inverts a matrix. This is a standard issue in using Fredholm-like equations for inverting conditional probabilities. However, since this method seems deprecated, no further discussion is included herein.

The one method that is available is `shareEnergySimple`, which is very similar to the `shareEnergySMD` method, except that the SMD energy is replaced with an expected contribution to the tower energy. This expectation is determined using splines and the `eeTowerFunction` [32]. The optimization of the parameters for the splines appears to have been done by Jason Webb.

6.3 Barrel

As the barrel ignores that towers may overlap, no energy sharing scheme is used.

7 Making Particles from Decay Products

It seems all possible pairs of γ's are currently considered. Also, no discrimination is made as to whether the pair truly comes from the decay of a parent particle other than cutting on the invariant mass of the digamma, $M_{\gamma\gamma}$. It seems that both photons do not need to be in the same sector, (based on Ref. [33]) though there is a cut hard coded that the difference in ϕ-bins must be less than 10.

8 Combination of Sectors

In most cases, it seems no combination of sectors is used, other than that previously mentioned in Section 7. The `StEEmcTower` class [34] does return true for a tower being a neighbor if the η and ϕ bins are adjacent, regardless of sector boundary. However, the class also includes the option for adding towers to the list of neighbors. Although `StEEmcTower` has some functionality across sections, one must also consider how exactly this class is used by other classes (i.e. the ones setting its neighbors).

No code uses the fact that there exists regions near sector boundaries where 3-layers of SMD are present, nor does any code allow for the SMD point to lay in a different sector than the strips. For example, it is noted in `EEmcSmdMap` [35]

```
* Limitations:
*
* 1. Strip ranges are approximate and should not be trusted to more
*    than two or three strips.  This should be sufficient for most
*    purposes.
*
* 2. Only U-V planes which are nominally within a given sector are
*    considered.  In other words, this class does not currently
*    provide any information on strips which overlap from adjacent
*    sectors near the tie-rods.
```

9 Particle Identification

9.1 End Cap

By particle identification, I mean the discrimination between leptons and photons. As of date, I can find no information on this item relevant to the EEMC. One could also consider discrimination between photons from decays verses those from other processes. No work has been done on this item either. Also, one can consider discriminating against jets (using some TPC information) or hadrons (using the post-shower).

9.2 Barrel

The only relevant methods I could find are the methods used by Mike Betancourt to distinguish his signal "prompt photons" from the background (other photons). In the barrel, the TPC can be used to distinguish between neutral and charged tracks. Mike uses Gaussian processes with a squared exponential kernel [36, 37] to estimate a function

$$y(\boldsymbol{x}) = \begin{cases} +1 & \text{if event is signal} \\ -1 & \text{if event is background} \end{cases} \tag{4}$$

where $\boldsymbol{x}$ is an array of possible relevant information for a given event. One advantage to using Gaussian processes is that the "hyper-parameters" show which elements of $\boldsymbol{x}$ are most relevant.

References

[1] See `http://www.star.bnl.gov/protected/spin/balewski/talks/2004-EEMC-Soft-Poster.ppt`

[2] See `http://www.star.bnl.gov/protected/spin/balewski/talks/2004-EEMC-soft-meeting.ppt`.

[3] See[6] `CVS://StRoot/StGammaMaker/StBarrelEmcClusterMaker.cxx?rev=1.5`

[4] See `http://www.star.bnl.gov/public/eemc/simulations/MineSweeper/`.

[5] See `CVS://offline/users/aliceb/StRoot/StEEmcPool/StEEmcIUPi0/StEEmcIUClusterMaker.h?rev=1.1` and `CVS://offline/users/aliceb/StRoot/StEEmcPool/StEEmcIUPi0/StEEmcIUClusterMaker.cxx?rev=1.1`.

[6] See `CVS://offline/users/aliceb/StRoot/StEEmcPool/StEEmcClusterMaker/StEEmcClusterMaker.h` and `CVS://offline/users/aliceb/StRoot/StEEmcPool/StEEmcClusterMaker/StEEmcClusterMaker.cxx`.

[7] See `http://drupal.star.bnl.gov/STAR/node/14867`.

[8] See `CVS://offline/users/aliceb/StRoot/StEEmcPool/StEEmcClusterMaker/StMyClusterMaker.h` and `CVS://offline/users/aliceb/StRoot/StEEmcPool/StEEmcClusterMaker/StMyClusterMaker.cxx`.

[9] See `CVS://offline/users/aliceb/StRoot/StEEmcPool/StEEmcClusterMaker/StAbClusterMaker.h?rev=1.1` and `CVS://offline/users/aliceb/StRoot/StEEmcPool/StEEmcClusterMaker/StAbClusterMaker.cxx?rev=1.1`.

[10] See `http://www.star.bnl.gov/protected/spin/balewski/2006-pi0-towerOnly/0-600K-minB-eve/` as well as `http://www.star.bnl.gov/protected/spin/balewski/talks/2004-Tower-Only-pi0-Finder.ppt`

[11] See `http://en.wikipedia.org/wiki/DBSCAN`.

[12] See `http://en.wikipedia.org/wiki/K-means_clustering`.

[13] See `http://drupal.star.bnl.gov/STAR/system/files/AlicePionCodeDocumentation.pdf`.

[14] See `http://en.wikipedia.org/wiki/Cluster_Analysis#Fuzzy_c-means_clustering`.

[15] See `http://drupal.star.bnl.gov/STAR/blog/kkrueger/2011/jun/06/brief-report-pi0-reconstruction-studies-summer-2010`.

[16] See `http://drupal.star.bnl.gov/STAR/blog-entry/ahoffman/2007/nov/07/stgammapointmaker`.

[17] See `CVS://StRoot/StSpinPool/StGammaPointMaker/Attic/`.

[18] See `CVS://StRoot/StPreEclMaker`.

[19] See `CVS://StRoot/StPreEclMaker/StEmcOldFinder.h?rev=1.2` and `CVS://StRoot/StPreEclMaker/StEmcOldFinder.cxx?rev=1.3`.

[20] See `http://drupal.star.bnl.gov/STAR/subsys/bemc/software/clustering/old-algorithm`.

[21] See `CVS://StRoot/StGammaMaker/StGammaFitter.h?rev=1.6` and `CVS://StRoot/StGammaMaker/StGammaFitter.cxx?rev=1.7`

[22] See `CVS://StRoot/StGammaMaker/StGammaCandidateMaker.h?rev=1.10` and `CVS://StRoot/StGammaMaker/StGammaCandidateMaker.cxx?rev=1.25`.

[23] See `CVS://StRoot/StEEmcUtil/StEEmcSmd/EEmcSmdGeom.cxx?rev=1.16`.

[6] All `CVS://` in the References section should be replaced with `CVS://`

[24] See CVS://offline/users/aliceb/StRoot/StEEmcPool/StEEmcIUPi0/StEEmcIUPointMaker.h and CVS://offline/users/aliceb/StRoot/StEEmcPool/StEEmcIUPi0/StEEmcIUPointMaker.cxx.

[25] See, e.g. http://drupal.star.bnl.gov/STAR/blog/kkrueger/2009/08.

[26] See CVS://offline/users/aliceb/StRoot/StEEmcPool/StEEmcPointMaker/StMyPointMaker.h and CVS://offline/users/aliceb/StRoot/StEEmcPool/StEEmcPointMaker/StMyPointMaker.cxx.

[27] See CVS://offline/users/aliceb/StRoot/StEEmcPool/StEEmcPointMaker/StAbPointMaker.h and CVS://offline/users/aliceb/StRoot/StEEmcPool/StEEmcPointMaker/StAbPointMaker.cxx.

[28] See CVS://StRoot/StEpcMaker/StPointCollection.cxx?rev=1.28.

[29] See CVS://StRoot/StEpcMaker/.

[30] See http://drupal.star.bnl.gov/STAR/subsys/bemc/software/clustering/point-maker.

[31] See http://drupal.star.bnl.gov/STAR/blog-entry/jwebb/2008/mar/11/work-progress-conversion-method and following blogs.

[32] See CVS://StRoot/StEEmcPool/StMaxStripPi0/eeTowerFunction.cxx?rev=1.2.

[33] See CVS://StRoot/StEEmcPool/StEEmcPi0Mixer/StEEmcPi0Maker.cxx?rev=1.5.

[34] See CVS://StRoot/StEEmcPool/StEEmcA2EMaker/StEEmcTower.h?rev=1.5 and CVS://StRoot/StEEmcPool/StEEmcA2EMaker/StEEmcTower.cxx?rev=1.3.

[35] See CVS://StRoot/StEEmcUtil/EEmcSmdMap/EEmcSmdMap.h?rev=1.10

[36] See http://drupal.star.bnl.gov/STAR/system/files/gaussianProcess.pdf

[37] See http://drupal.star.bnl.gov/STAR/system/files/betancourt2010.12.16.pdf