The Weighted Analysis Technique
Review + Implementation for Milagro
Conceptual Foundation

\[ p_i(\vec{k}) = p(\vec{k} | \vec{k}_i, \psi_i) = PSF(\vec{k} - \vec{k}_i, \psi_i) P_\gamma(\psi_i) \]

\[ w(\vec{k}) = \sum_{i} p_i(\vec{k}) = \sum_{i} PSF(\vec{k} - \vec{k}_i, \psi_i) P_\gamma(\psi_i) \]
Source Detection

\[ P(\bar{w} \geq \bar{w}_{obs} \mid g(\bar{w}, N_{exp}, \bar{k})) \]
Source Detection with Truncated PSF
\ln(N_{obs})

\bar{W}_{obs}
Further Approximations

\[ \sum_{N=N_{obs}}^{\infty} P(\bar{w} \geq \bar{w}_{obs} | g(\bar{w}, N, \tilde{k})) P(N | N_{exp}) \]

\[ P(\bar{w} \geq \bar{w}_{obs} | g(\bar{w}, N_{obs}, \tilde{k})) P(N \geq N_{obs} | N_{exp}) \]
Sensitivity

• We’d like to compare optimal binned, maximum likelihood and weighted analyses, but in general we need an application specific Monte Carlo.

• We can obtain analytic solutions for optimal bin vs. the weighted analysis for the case of Gaussian PSFs, a hard background cut, and the limit of large statistics.

• Sensitivity has form of $AN_s/\sqrt{b}$, goal is to find $A$ which characterizes the sensitivity of the analysis.
### One Gaussian PSF

<table>
<thead>
<tr>
<th>Optimal Bin</th>
<th>Weighted Analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ \text{Signal} = \int_0^R \frac{N_s}{2\pi\sigma^2} e^{-r^2/2\sigma^2} 2\pi r , dr = N_s (1 - e^{-R^2/2\sigma^2}) ]</td>
<td>[ \text{Signal} = \int_0^\infty N_s \frac{1}{2\pi\sigma^2} e^{-r^2/2\sigma^2} \left( \frac{1}{2\pi\sigma^2} e^{-r^2/2\sigma^2} \right)^2 2\pi r , dr = \frac{N_s}{4\pi\sigma^2} ]</td>
</tr>
<tr>
<td>[ A = 0.255/\sigma ]</td>
<td>[ A = 0.282/\sigma ]</td>
</tr>
</tbody>
</table>

**Weighted Analysis is \sim 10\% More Sensitive**
Two Gaussian PSFs

25% of events from PSF 1/3\(\sigma\) wide
and 75% from PSF 1\(\sigma\) wide.

Weighted Analysis is \(~56\%\) More Sensitive than Optimal Binned Analysis
Sensitivity Advantage Depends on Spectrum of PSFs

**Optimal Bin Signal**

\[ N_s \int_0^R PSF(\sigma, r)g(\sigma)2\pi r \, dr \, d\sigma \]

**Weighted Analysis Signal**

\[ N_s \int \int PSF(\sigma, r)PSF'(\sigma, r)g(\sigma)2\pi r \, dr \, d\sigma \]

- Sensitivity depends on shape of composite PSFs.
- Weighted Analysis reduces to a binned analysis if a top-hat weighting function is used. The true PSF is the optimal weighting function (Woodhams) and the sensitivity of the weighted analysis greater than or equal to a binned analysis.
- Errors in PSF determination do affect the PSF sensitivity.
Sensitivity Summary

• Using $P_\gamma$ enhances the sensitivity advantage of the weighted analysis technique over a binned analysis.

• Alexandreas et al. performed a Monte Carlo to compare optimal binned and maximum likelihood for the case of a single Gaussian PSF, a hard background cut, and large statistics, and found a ~10% advantage from maximum likelihood. This is the same as the advantage from the weighted analysis technique (first example), and suggests that weighted analysis is similar in sensitivity to well-implemented maximum likelihood in at least some circumstances.
Application to Milagro:

Library Exists to Implement the Weighted Analysis Technique

• Robust, extensively tested library.

• All of the difficult implementation details are coded, and custom analyses can be easily created.
Example for Steady Source Search

```c
int main()
{
    SkyMap *totalSkyMap;
    LocalMap *tempLocalMap;
    BackgroundMap *tempBackgroundMap, *totalBackgroundMap;
    NSArray *photonArray;

    //Initialize
    //Create classes, tell Map classes which PSFs to use and load precalculated
    //probability tables. Tables can be shared, initialization is ~10 lines of code

    while(!hellFrozenOver)  //Main loop
    {
        //Your code to read in data blocks (2 hour blocks?),
        //filter, determine PSF and P_gamma for each event and pack into eventArray

        //Put events on sky and local maps
        [totalSkyMap addPhotonArray: eventArray];
        [tempLocalMap addPhotonArray: eventArray];

        //Create background set from local map, add to total background
        tempBackgroundMap = [tempLocalMap integratedBackground];
        [totalBackgroundMap addSkyMap: tempBackgroundMap];
    }

    //Find Significance at a set of points
    myLocations = [totalBackgroundMap searchThePoints: myLocations WithSignal: totalSkyMap];
    //To search the entire sky for interesting points
    hotSpots = [totalBackgroundMap completeSearchForProbabilityBelow: -2 WithSignal: totalSkyMap];
}
```
GRBMain
- Initializes the GRBMaster object and schedules when reports will be archived - this is the main process for the GRBSearch program, with GRBReadData and GRBHunters running as sub-threads.

GRBReadData
- This process is responsible for reading data from the disk, applying cuts, determining the PSF and weight to use, identifying and removing data errors. When the events are given to GRBMaster they will all be in order and "good" events. If a bad block of data is identified a reset message is sent to GRBMaster to start a reinitialization of the search.

GRBMaster
- During Startup starts all 9 of the GRBHunter threads, the SignalResponder program if it is not already running, and the ReadGRBData thread to read in the information.
- During normal running, this thread is responsible for adding showers to the signal and local maps in 20 second intervals, producing the background map, and handing the signal and background maps to the first GRBHunter (40 second search). This thread also organizes the archiving of run histograms, startup/shutdown, and other run management.

GRBHunter - 40 sec.
- Each of the 9 hunters adds 2 sky and background maps, and searches the resulting map for a signal. Whenever a unique composite map is created it is given to the next GRBHunter. If any hunter finds a signal below a threshold probability, it sends a message with the position and time to the SignalResponder program.

GRBHunter - 80 sec.
GRBHunter - 160 sec.
GRBHunter - 320 sec.
GRBHunter - 10.7 min.
GRBHunter - 21.3 min.
GRBHunter - 42.7 min.
GRBHunter - 85.3 min.
GRBHunter - 170.7 min.

Signal Responder
- The separate SignalResponder program collects the signals identified by the GRBHunters. Since a strong signal will be identified in multiple searches, they are grouped and sorted to form one notification. If the signal crosses a set threshold a GRB alert is sent out, triggering the saving of raw data and alerting me and the current shift person.

E-mail Alerts

Legend
- Program
- Object
- Information Flow

Current Medium Duration
GRB Search
Weighted Analysis Library

Pros:
• Enhanced sensitivity over binned analyses
• Stable, fast library code
• All key functionality has been implemented and fully tested
• Custom implementations are easy
• Many advanced features such as distributed processing, multi-threading, freeze-dried archives, dynamic interaction, and error recovery are very easy to implement

Cons:
• Analysis complexity
• PSFs, background weighting and probability calculations must be carefully considered
• Learning curve for using the library
• Requires additional packages to be installed