CDF/PUB/CDFR/PUBLIC/8015
Draft 1.0
September 24, 2006

Space-Time Vertex Reconstruction Using COT Tracks

M. Goncharov[1], V. Krutelyov[2], D. Toback, P. Wagner

Texas A&M University

Abstract

This note describes a space-time vertex reconstruction using COT tracks. Studies of its performance using data indicate that it does not introduce any timing or Z position measurement bias and has resolutions of 0.4 ns in time and 0.2 cm in Z on average for the sample of $W \to e\nu$ events. We find that we have good vertex separation and that the efficiency for events with high track multiplicity is 100%.

[1]maxi@fnal.gov
[2]At University of California at Santa Barbara since Feb 2006

Contents

1 Introduction

Each collision within the CDF detector happens at some specific place and time. The standard vertexing [1] currently only measures the position of the primary collision. The measurement of position is crucial for the sinθ measurement which is part of the E_T measurement that determines the kinematics of electrons, photons, jets etc. If the luminosity is high enough two or more collisions can occur, some of which can be very close to each other in space and/or time, and can cause event measurement problems. In the case where the two collisions are close in Z this can cause a mis-measurement in the vertex position. In general this will have a negligible effect on sinθ and thus E_T. If the time of the collision is important, separating between two collisions that are well separated in time but not space becomes crucial; otherwise a reported collision time can be poorly measured and/or can have a significant bias. An example analysis where such a measurement is important is the search for heavy, long-lived particles [2, 3]. Thus, there are advantages to separate vertices that might otherwise be combined and to provide a high quality measurement of the time and position.

In this note we present a new vertexing algorithm that takes advantage of the track Z position (Z_0) and track time (T_0) for each good track found in the COT, and does clustering in both the space and time dimensions. To first order there is no correlation between the space and time dimensions for the collisions [3] and the RMS of the distribution along the beam line (Z axis) is $\simeq$30 cm, while the width of the time distribution is $\simeq$1.3 ns. This is shown in Fig. 1 (N.B. we are showing the vertexing results from a data sample of $W \rightarrow e\nu$ events before description of the algorithm for pedagogical reasons). Since the COT track Z_0 and T_0 resolution are small compared to the RMS of the parent distribution [4], it becomes clear that we have the capability to separate vertices that are well separated on the track resolution scale, either in time or space. Fig. 2 shows this schematically where we consider events with two collisions using the standard vertexing and our new algorithm (again here assuming the results of our later algorithms for pedagogic purposes). By vertexing in two dimensions we expect that separation of vertices that are distanced in either space or time will help our analysis. It is crucial for the long-lived particles searches to separate collisions that are far apart in time. Figure 3 shows the difference between the electron track T_0 and the ZVertex average time. The secondary peaks are the results of combining two collisions that have very different track multiplicity and that are far away from each other in time.

Section 2 describes the procedure of turning sets of tracks from the COT into a set of vertices using a new clustering algorithm. In particular, we use a Expectation Maximization (EM[4]) technique [5] to find the parameters of our vertices, and to take into account that there is potentially more than one collision in the event using simple merging and splitting techniques to determine the true number of collisions. Note that throughout the text we will refer to the reconstructed vertex as a "cluster", while we

[3] Note that this is not completely true, as will be discussed in Section 4.2.

[4] Do not confuse with the EM calorimeter!

will refer to the "vertex" to denote the true underlying collision. In Sections 3 we show how well our clustering works by giving the reconstruction efficiency, the position and time resolutions, as well as our ability to separate vertices. In Section 4 we conclude with some examples of how well our vertexing works for certain types of measurements.

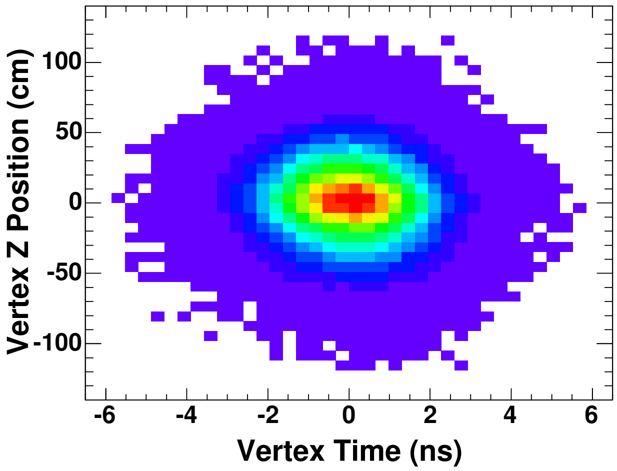

Figure 1: The plot shows the space-time distribution for vertices for a sample of $W \rightarrow e\nu$ data. One can see that the two dimensions are, to first order, orthogonal to one another. We note that these results presuppose the results shown in later sections for pedagogical reasons.

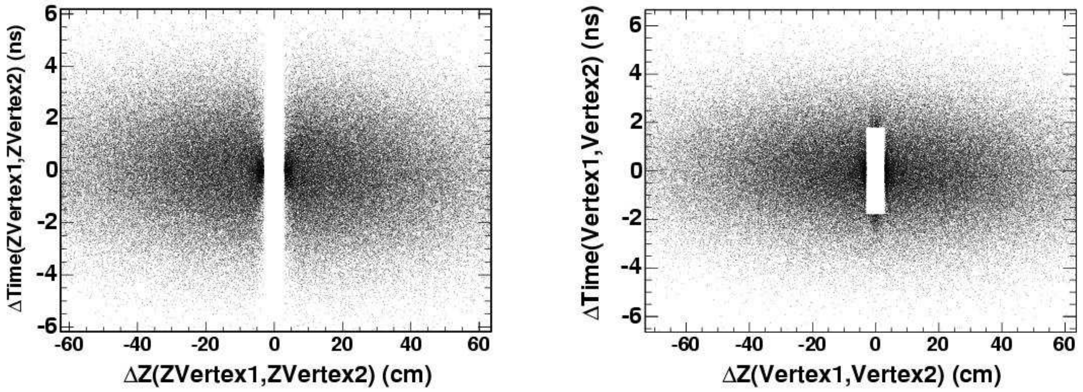

Figure 2: This plot shows the space-time differences from two vertices in the sample of $W \to e\nu$ events described in Fig. 1. On the left: The standard ZVertex algorithm which does not use timing information. On the right: Our clustering algorithm that uses both space and time information and partially recovers cases when collisions are separated only in time. We note that these results presuppose the results shown in later sections for pedagogical reasons.

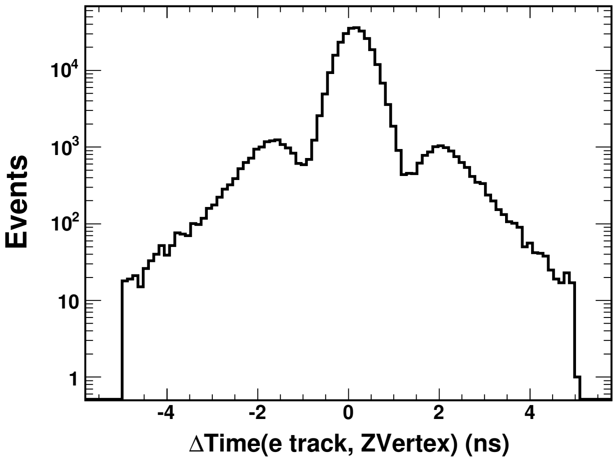

Figure 3: This plot shows the difference between the electron T_0 and ZVertex time. The additional peaks in the distribution appear due to the specifics of vertex T_0 calculation algorithm. In particular, in the case of two collision vertices close in Z being combined by ZVertex: if the true vertices are significantly separated in time, the calculated T_0 for the combined vertex will be closest to the time of only one of the vertices.

2 Vertex Reconstruction

2.1 General Description

In this section we describe our vertex reconstruction techniques. We begin with the description of the general ideas behind the methodology, and then we continue with the description of the clustering formalism and our particular implementation. It can be split into three separate ideas: 1) The defining of our list of clusters, 2) the determination of the best-fit values for the cluster parameters using the EM technique, and 3) adjusting the number of vertices by merging and splitting clusters if necessary. At the core of the EM algorithm lies the assumption that all good tracks are from some underlying parent vertex distribution with a particular mean and resolution for the Z_0 position and time values. Our job is then to develop algorithms that determine the correct number of vertices, and to measure the properties of each. For pedagogic reasons we begin with a description of the EM portion of the algorithm with the assumption that we have correctly picked the right number of vertices, and combine the description of the initial list of clusters, merging and splitting to the second section.

2.2 Expectation Maximization (EM) Formalism

The EM procedure is premised on the idea that a track has a finite probability to be a member of any particular vertex. Thus, the goal of the clustering is to maximize the overall probability or likelihood of the data being well described. This is crucial in our case because tracks in the COT are not always from a particular collision. In particular, there can be a fake track, or a real track from a comic ray. Formally speaking, instead of assigning cases or observations to clusters to maximize the differences in means for continuous variables (Z_0 position and time in our case), the EM clustering algorithm computes probabilities of cluster memberships based on one or more probability distributions. The goal of the clustering algorithm then is to maximize the overall probability or the likelihood of the data, given the (final) clusters. Said differently, we use the simple assumption that all points one wishes to cluster come from some underlying distributions with well defined mean and resolution values. The question: "what vertex does a particular track come from?" should be replaced with "what is the probability that a given track originated from a given vertex?". At the end, each vertex is assigned an overall weight that reflects what fraction of all tracks probably originated from it. Fig. 4 illustrates the point using a 1-dimensional example with two vertices. Clearly it is impossible to say to which vertex points in the middle belong to, but it is intuitively obvious that vertex A accounts, let's say, for 40% of all points, while vertex B produced only 60% of the points. At the end each vertex is assigned mean and RMS values, which intrinsically implies that the vertex generates points according to a Gaussian probability distribution function (PDF):

$$PDF = \frac{1}{(2\pi)^{N/2}\|M\|^{1/2}} exp[-\frac{1}{2}(x-\mu)^T M^{-1}(x-\mu)], \tag{1}$$

where M is a covariance matrix, μ is the vector of mean values, and x is the vector representation of each point (or track). In our case N=2, one dimension is the Z_0 position of the tracks, and the other dimension is the time of the track.

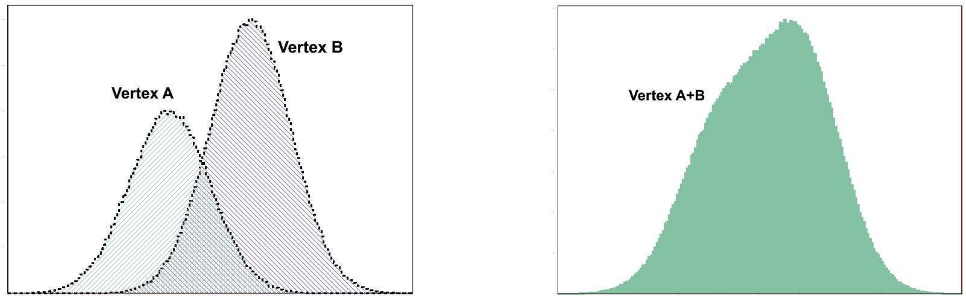

Figure 4: On the left: the true distribution for two vertices close to each other. Vertex B is responsible for 60% of the points, while vertex A generated 40% of all points. On the right: the resulting distribution of points one might observe in the data. Note that it has larger width than the PDF that originally generated the vertices.

The next step is the maximization procedure which is equivalent to a likelihood fit. We assume a set of parameters, our state λ, and calculate the likelihood. We then vary the parameters to find the best likelihood and stop when we find that attempting to optimize further doesn't change the values of the parameters by more than a percent. We start by denoting our current knowledge of the problem at the iteration step t in the form of:

$$\lambda_t = (\vec{\mu},\ M,\ \vec{p}),$$

where $\vec{\mu}$ are the mean values for the generating PDF (Equation 1), M is the covariance matrix, and $\vec{p}$ is a vector of weights assigned to each model (vertex). Again, we assume for now a number of vertices and assign initial parameters to the PDF function based on the *a priori* knowledge. Vertices can be assigned equal weights in the beginning, or in any other way, the only requirement is that the sum of all weights must be equal to one. At this point λ_1 is defined and the iterative procedure can be started.

Let's denote by w_i vertex number i. Our goal is to find the optimal values for the parameters of all vertices to describe data points, the collection of which we will denote as $\vec{x}$. The algorithm is split into two steps. At first step, called E-step, we calculate expected values for every data point:

$$P(w_i|x_k\lambda_t) = \frac{p(x_k|w_i\lambda_t)P(w_i|\lambda_t)}{p(x_k|\lambda_t)} = \frac{p(x_k|w_i,\mu_i,M_i)p_i(t)}{\sum_{j=1}^{NM} p(x_k|w_j,\mu_j,M_j)p_j(t)}, \tag{2}$$

where NM is the number of models (Gaussians), and $p(x_k|w_i,\mu_i,M_i) == PDF$ as described in Formula 1. The calculated value gives the probability for a given point to originate from a given vertex.

At the next step, the so called M-step, we adjust the estimated parameters for the generating distributions

$$\begin{aligned}
\mu_i(t+1) &= \frac{\sum_k P(w_i|x_k\lambda_t)x_k}{\sum_k P(w_i|x_k\lambda_t)}, \\
M_i(t+1) &= \frac{\sum_k P(w_i|x_k\lambda_t)[x_k-\mu_i(t+1)][x_k-\mu_i(t+1)]^T}{\sum_k P(w_i|x_k\lambda_t)}, \\
p_i(t+1) &= \frac{\sum_k P(w_i|x_k\lambda_t)}{NM},
\end{aligned} \tag{3}$$

where index i runs over all models (vertices), index k runs over data points we are trying to cluster, and index t is the iteration number. The first equation calculates the mean values for each vertex, the second one calculates the covariance matrix, and the third one evaluates the weights that are assigned to each vertex. We stress out that index i denotes the vertex (model) number, so M_i is the calculation of the full covariance matrix for the vertex i, and $[x_k-\mu_i(t+1)][x_k-\mu_i(t+1)]^T$ is the $N\times N$ (2×2 in this case) dimensional matrix. In general the calculation stops when the change in the model parameters is below the user defined thresholds; we choose it to be that all the mean values for each vertex changes by less than 1%. The described procedure amounts to a likelihood fit but prescribes an iterative approach to accomplish the same goal that the maximum likelihood would - to provide the best data description by adjusting vertex parameters.

2.3 From Initial Conditions to the True Number of Vertices

There are many different techniques that people use to estimate the true number of vertices [6]. The primary pitfalls are cases where two collisions are merged into a single cluster, and cases where a single collision is split into two clusters. We use a simple set of algorithm decisions to remove these pathologies within our resolution and show in the next section that our choices are robust. Algorithmically, we establish an initial guess for the number of collisions that overestimates the number of vertices and merge if the means of two clusters are within the expected RMS of a single cluster.

We begin by doing loose cluster seeding. In particular we create a track list with all good tracks ordered by P_T. The highest P_T track then becomes a seed and we look to add tracks to this seed. All tracks within one expected (*a priori*) RMS from its mean value, in both Z and time, are added to this cluster and removed from the track list collection. The next highest track remaining on the list then becomes the second seed, and tracks are added in the same way. This process is continued until there are no tracks left. The EM iterative procedure described above then starts.

Since we have done an initial seeding in a way that is likely to overestimate the number of clusters, this should make splitting unnecessary. We next concentrate on merging. If during the EM procedure the means of two clusters are pushed by the algorithm too close to each other[5] (i.e., so that it is likely actually one cluster), one

[5] The value we have chosen equals to one *a priory* known RMS.

cluster is killed, and the clustering described above starts over again i.e., the parameters are determined for the λ_1 state and the EM iterations can start. Similarly, if two clusters share 100% of the tracks one of the vertices is deleted from the cluster list, the remaining cluster is then able to readjust it's weight. Again the EM procedure is restarted.

While there might always be clusters that are a combination of two true vertices, we make no further attempt to split them. An example case is shown in Fig. 4. This could, in principle, be determined because the resulting RMS is too large to be consistent with the expected resolution for a single collision. Unfortunately, the criteria for deciding what is "too close" is a balancing act between allowing too many vertices to be merged and the possibility of splitting a single vertex with the potential to lose efficiency to create a good cluster in the first place. We did investigate a splitting procedure, by creating two clusters from single clusters that have abnormally large RMS, and running the EM clustering one more time. For technical reasons this part of the procedure did not work as expected and did not improve things, in particular we lost good vertices in a non-negligible fraction of events. In order to confirm that we are in a region which is useful for analysis we studied the effect of keeping a loose (but not too loose) merging requirement and note that it maintains an excellent efficiency and negligible impact on our performance. This is the topic of our next section.

3 Algorithm Performance

We next study the performance of our vertexing algorithms. In particular, we are interested in understanding if there are biases in our results, as well as determining the resolution and efficiency of our reconstruction. To do this study we use an inclusive sample of $W \to e\nu$ events from the data. The data sample used in this study comes from the W_notrack trigger (bhel0h dataset) starting from run 190851. To reduce bias and increase our statistics we only require that the electron pass standard ID cuts beyond the trigger requirements. The whole data sample corresponds $\simeq$ to 370 pb^{-1} of integrated luminosity. Note that in order to properly measure these quantities we remove potentially problematic tracks from our vertexing by using the requirements in Table 1, and we remove the electron track from the track list so it can be used as a tool for measuring our performance. We also note that in all plots, we require 4 or more good tracks in every vertex except where otherwise noted. Our primary vertex is always selected to be the vertex with the highest $\sum P_T$ (we have removed the electron track so as to not bias ourselves to always picking that vertex).

The distributions in Z and time for our sample are shown in Fig. 5. The plots show that the collisions happen on average at zero in Z position and time with an expected RMS. The non-Gaussian enhancement in the Z position around zero is due to the algorithm combining two vertices into one cluster if the collisions are close to each other. This feature is described in more detail in the following sections. The falloff at $|Z| > 70$ cm is due to the tracking requirement in Table 1. We next look for potential biases in the clustering itself.

$P_T > 0.3$ GeV
$\lvert Z \rvert < 70$ cm
Track rapidity $\lvert\eta\rvert < 1.6$
N COT Stereo Layers(5 hits) > 2
N COT Axial Layers(5 hits) > 2
0.05 ns $<$ COT T_0 Error < 0.8 ns
Z position Error < 1 cm
Not a slow proton

Table 1: The requirements used to select high quality tracks for use in clustering. The definition of a low proton track is given in Table 2.

Charge > 0, $P_T < 2.0$ GeV, dEdX > 22

Table 2: Definition of a slow proton track.

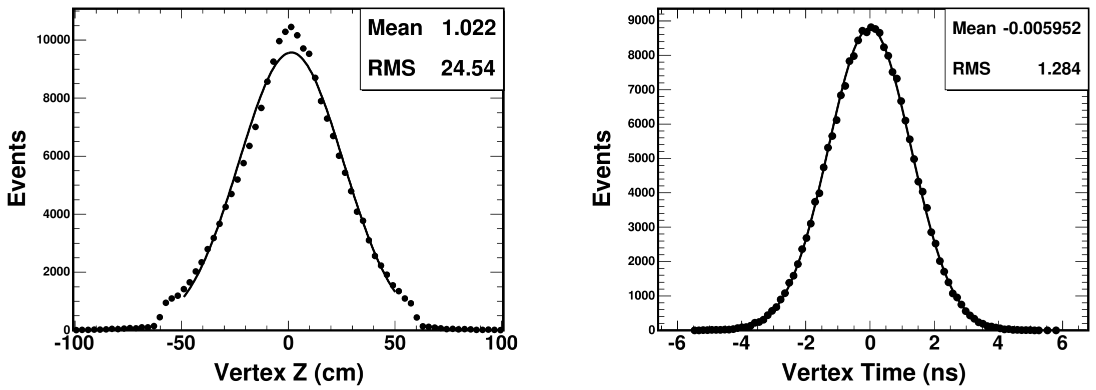

Figure 5: The reconstructed Z and time for the vertexes for our sample of $W \rightarrow e\nu$ events. One can see that the distributions are centered at zero and have expected Gaussian shapes and RMS values. Z distribution is not a perfect Gaussian because the algorithm merges vertices that are close to each other.

3.1 Artificial Bias Check

Since the electron track does not participate in the clustering we can use it as a way of searching for vertexing biases in the time or Z measurement. Fig. 6 shows the ΔZ and Δt distribution between the electron track and vertex measurement. There is no evidence for a bias introduced by the algorithm. As a second check, we note that the resolution in Z position is different for the electron tracks with silicon hits and for those without silicon hits. In the first case the ΔZ distribution is dominated by the vertex resolution (albeit with lower statistics), and in the second case the resolution is dominated by the track itself. We point the reader to the second Gaussian in both plots. These are a class of events where there are two collisions and the vertex under investigation (the highest ΣP_T vertex, not including the electron track again) is not the same vertex as the one electron originates from. To confirm this we note that the RMS of the secondary Gaussian is ≈ 25 cm which is the equivalent of the RMS of all collisions. We note that all the above comments hold as well for the timing distribution in Fig. 6.

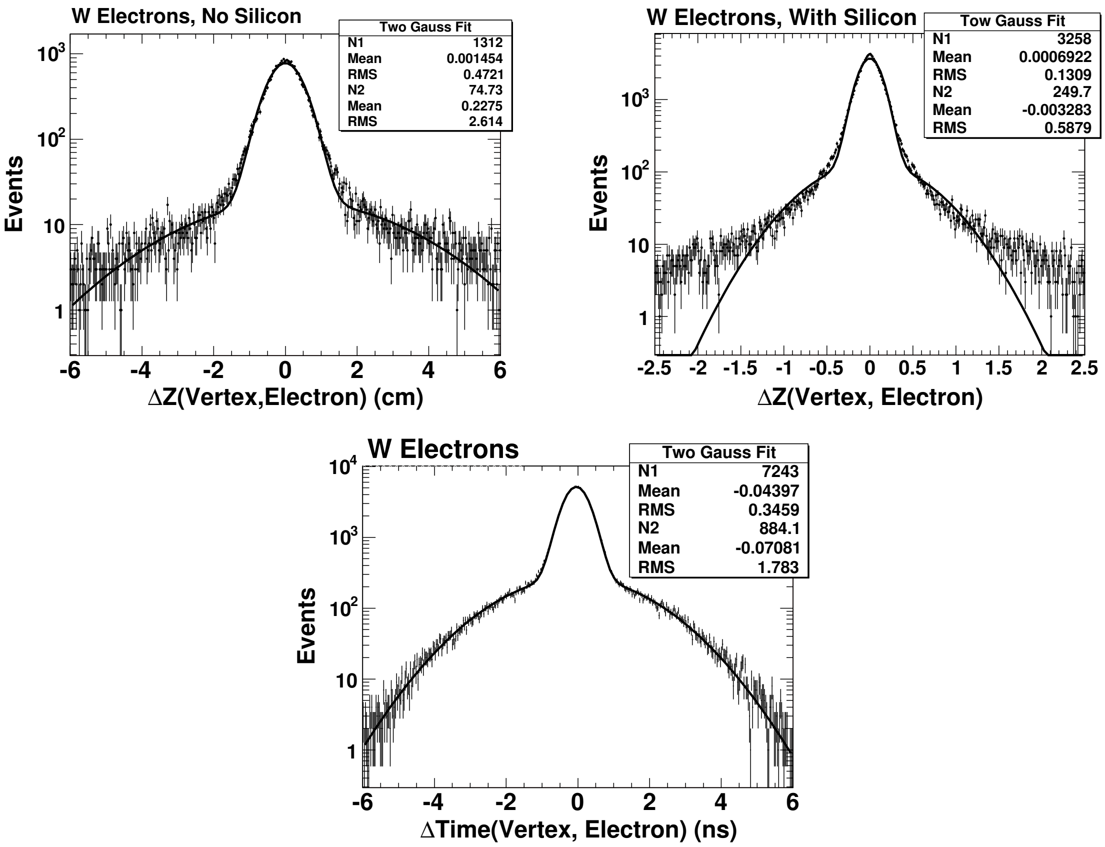

Figure 6: The difference between the highest $\sum P_T$ cluster and electron track in the sample of $W \rightarrow e\nu$ events. The electron track does not participate in the vertexing. The width for the Z distribution is different when one takes electrons with or without silicon. All plots show there are no artificial biases.

3.2 Resolution

We estimate the resolution by selecting events with only one well-reconstructed vertex, dividing the clustered tracks into two random groupings, reconstructing each group separately and comparing. Fig. 7 shows the difference of the means of the two clusters divided by $\sqrt{2}$ to take into account the two measurements. We divide that difference by the reported resolution as a way of seeing how well the algorithm reports its resolution. We note that in both cases the clusters underestimate the true error by $\sim$20%.

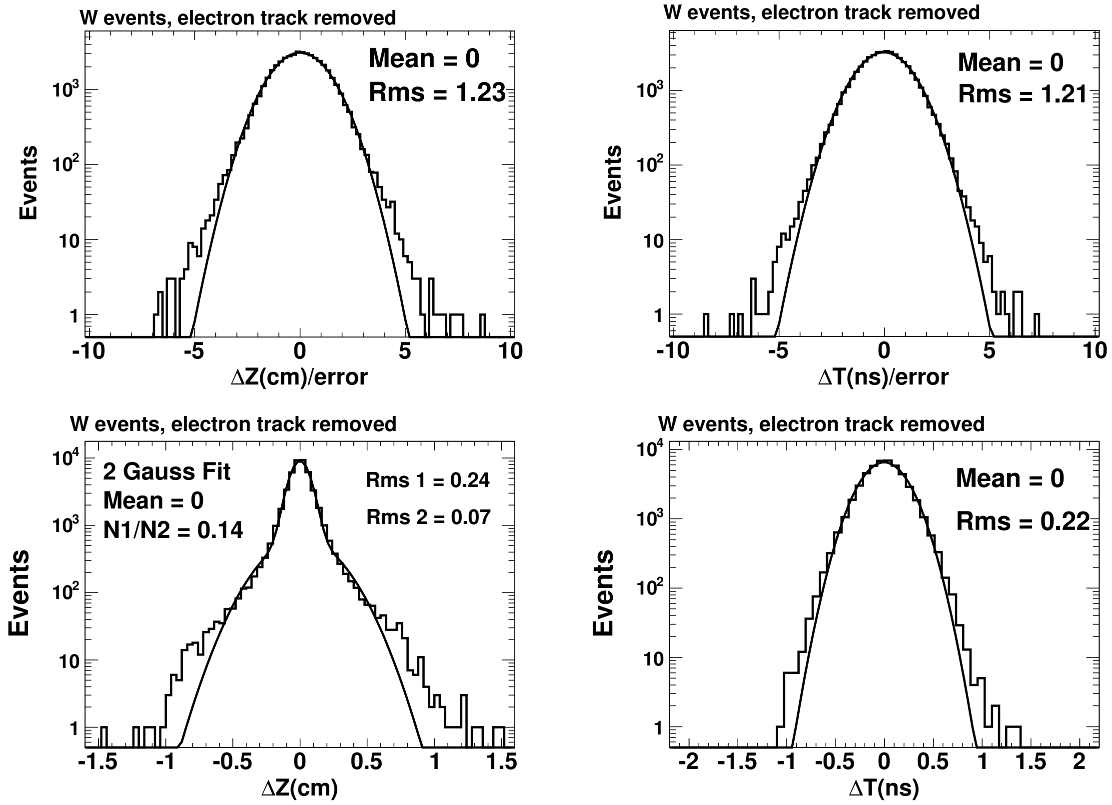

Figure 7: Plots on top show the differences between two clusters from the same vertex normalized to the errors reported by the clusters. Cluster errors are underestimated by 20%. Plots on the bottom show the unnormalized differences between clusters. This is the true resolution of the clustering algorithm. Please note that the factor of $\sqrt{2}$ is already taken out.

The resolutions reported by the clusters themselves are shown in Fig. 8. The error on the mean drops as expected like a square root with the number of the tracks in the vertex. We point the reader to the second peak on the plot for the Z position (Fig. 8 left) at $Z \approx$0.8cm. This indicates that there are cases when two different vertices are combined in one cluster, and this might have an effect on our analysis. To follow up on this issue we look at the difference between two clusters in Fig. 9 for events with two well reconstructed vertexes. The timing distribution shows no outstanding features, while

Z position shows a drop in the middle. That is consistent the previous statement that the algorithm does not separate vertices that are too close in Z, probably because the underlying true resolution of the vertices is not selected optimally, but is overestimated.

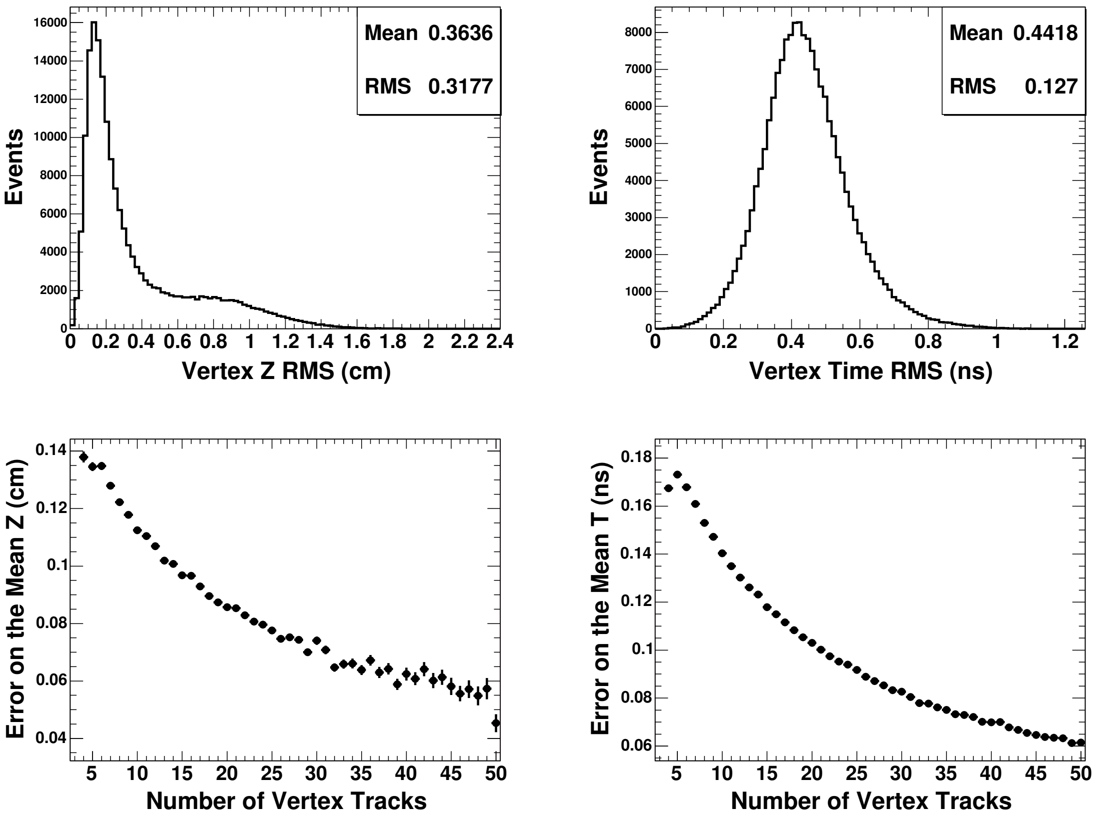

Figure 8: Top two plots show the resolution distributions reported by clusters. The second peak at $Z = 0.8$ cm in the top left distribution indicates that that the assumed true resolution is not selected optimally and there cases when two vertices close in Z are combined into a single cluster.

Adding two vertices is not a problem here. If two vertices are combined in Z and have a large RMS, in this case between 0.6 cm and 1.6 cm, this has a negligible effect on our kinematic calculations. The real question is whether these cases are such that the time is potentially mis-measured. We check this in Fig. 10 by investigating for biases or timing resolution degradation in the time difference between well measured electron tracks and vertices that are potentially the combination of two separate collisions. In particular we consider events with large RMS >0.6 cm, see Fig. 8, but only consider well measured electrons that are close to the measured vertex. There is no substantive difference between this plot and that of Fig. 6. We consider this issue no further and move to measuring our efficiency.

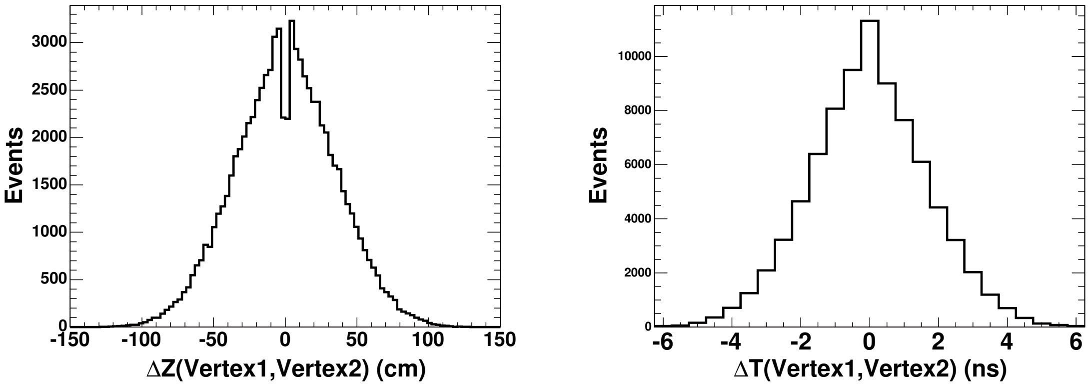

Figure 9: The plots show the measured differences between two clusters for events with multiple interactions. The dip in the middle of the Z position plot indicates that there are cases when two vertices close in Z are clustered as one. As noted in the text and shown in Fig. 10, this is not a problem for our timing measurement.

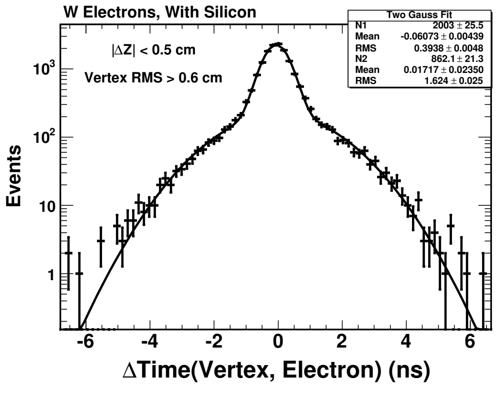

Figure 10: The time difference between well measured electron tracks and clusters that are potentially a combination of two separate collisions. In particular we consider events with large RMS >0.6 cm, see Fig. 8, but only consider well measured electrons that are close to the measured vertex. There is no substantive difference between this plot and that of Fig. 6 indicating no timing bias or resolution degradation for our "combined" vertices.

3.3 Efficiency

Finally we describe a number of ways of measuring the algorithm efficiency and checking our results. We begin by selecting events from our data sample with clusters with high track multiplicity, ensuring that there is a good vertex to be studied. We then take this high-track multiplicity cluster and look at various subsets of the tracks associated with it to see if they alone would be able to produce a good cluster using the same code. The efficiency can then be determined as the ratio of cases when a new vertex from this subset is found to all cases tried for this set of tracks. For example, the Fig. 11 shows the efficiency as a function of the number of tracks selected. The results for the lowest bins are given in Table 3 and show that the vertexing algorithm is over 90% efficient if 4 tracks are present (remember that our vertexing algorithm requires 4 or more tracks) and is nearly 100% efficient by 6 tracks.

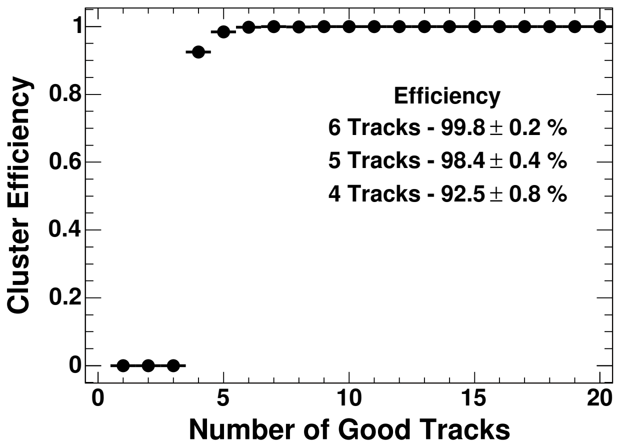

Figure 11: The vertex reconstruction efficiency as a function of the number of good tracks found from the collision. Tracks are selected at random from known good vertices with lots of tracks. After that vertexing is done only on selected tracks.

Number of Tracks	Efficiency	Error
4	92.5	0.8
5	98.4	0.4
6	99.8	0.2

Table 3: Direct efficiency measurement.

A second method looks at the efficiency of finding a vertex as a function of the track multiplicity by counting tracks inside a 2 cm x 2 ns window around the electron in our W events when we only consider events with one or fewer reconstructed vertices. This has the advantage of not biasing our result high by selecting events with a known good vertex. The disadvantage is that not all tracks are, for resolution reasons, in the window, giving us a known under-counting of the number of tracks from the collision (we allow all tracks in the event to contribute to creating the vertex). The results are shown in Fig. 12 as a function of number of tracks, ΣP_T of the tracks and the number of jets above 10 GeV. As expected, there is some efficiency for finding a vertex with 4 tracks, even though there are only "3" tracks in the window. However, the difference between the results in this efficiency method and the previous one are consistent, giving us confidence that the measurement is robust. We note for completeness that the efficiency as a function of sum P_T of all tracks plateaus at 5 GeV, and as expected, with one jet present the number of tracks increases, and the efficiency to find a cluster approaches 100%.

One should point out that the above numbers are somewhat sample dependent because the true vertexing efficiency is probably convoluted with the track quality[6]. For example, if a sample is chosen to be biased toward a higher average P_T of the tracks, the efficiency might be larger as a function of track number. Similarly, if the sample is chosen with only low P_T tracks, the efficiency would be much lower. To get a flavor of how track quality might affect our results Fig. 13 shows the efficiency, using the same method, as a function of good tracks, and the fraction of the good tracks for the $W \to e\nu$ sample. We note that only about 85% of the tracks in the sample pass the cuts, indicative that many of the tracks in the sample are probably not really ideally suited for our clustering as they are probably fake (i.e., the efficiency of the track requirements is very high). Said differently, the small difference between the results in the two figures, comparing the efficiency turn-on curve with good tracks only and with all tracks indicates further that the true turn-on curve is very sharp and is well measured. A threshold fit for Fig. 13 are listed in the Table 4 along with the statistical and systematic errors, and are very consistent with the results in Fig. 11.

Parameter	Value	Stat. Error	Syst. Error
Efficiency	99.8	0.6	0.2
Threshold	3.51	0.02	0.11
Width	0.62	0.03	0.40

Table 4: Efficiency for the number of tracks using the window method.

[6]Remember that only tracks passing the requirements in Table 1 are used in the vertexing.

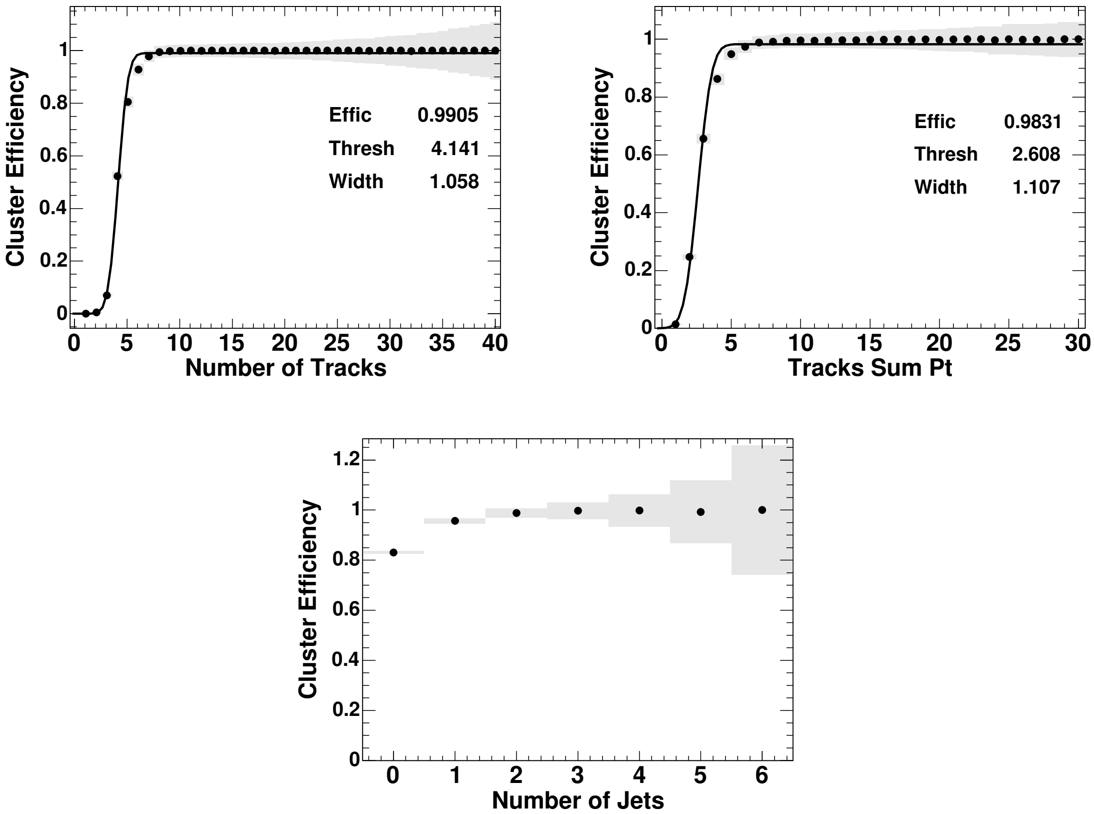

Figure 12: The cluster reconstruction efficiency as measured using the window method. The results are presented as a function of the number of tracks, tracks ΣP_T, and a number of jets ($E_T > 10$ GeV) in the event and are consistent with the results in Fig. 11.

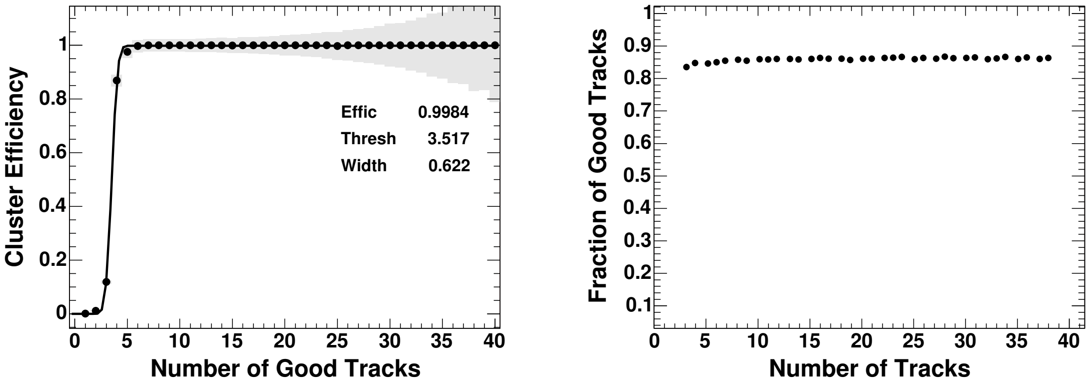

Figure 13: The cluster reconstruction efficiency measurement using the window method. On the left: Cluster reconstruction efficiency as a function of number of good tracks, On the right: fraction of good tracks as a function of the total number of tracks.

4 Discussion

Given the results in the previous section, we believe we have developed a powerful new vertexing method that gives a robust measurement of the Z and time of the primary collision with high efficiency if there are 4 or more tracks from the collision. We use the vertexing to look at a some physics quantities to further illustrate its power.

4.1 Luminosity Dependence

The average number of reconstructed vertices is expected to increase as a function of the instantaneous luminosity. As shown in Appendix A it turns out that this increase is linear. Fig. 14 shows our data results for the total number of reconstructed vertices as a function of the instantaneous luminosity measured by the CLC. We note that for most of the range the relationship is linear as expected. At high luminosity the curve starts deviating from linear. This can be understood that as more than one interactions occur, there is a higher and higher probability that interactions happen close to one another and are combined into a single reconstructed vertex. Thus, for luminosities up to 10^{32} our vertexing has enough power to separate vertices, on average, so as to not be noticeable on this scale. We note for completeness, that the minimum number of vertices in the plot is fixed at 1 because we are looking at W events, i.e., we have required a collision.

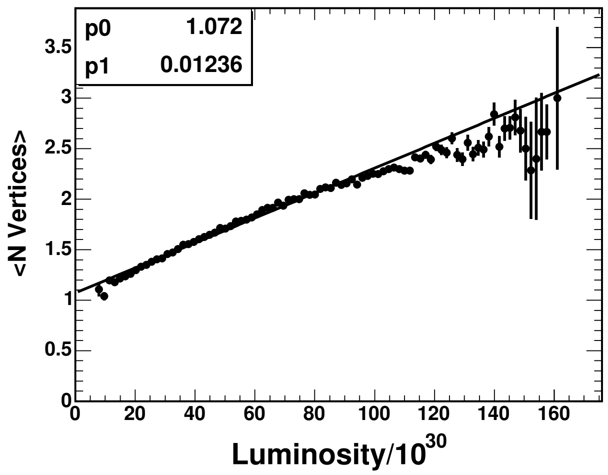

Figure 14: Number of reconstructed vertices as a function of instantaneous Luminosity. The events pass a trigger that is biased to select collisions.

4.2 Correlation between the Position and Time of Collisions

As previously noted, to first order the position and time of the collision are unrelated. However, this is not true in principle if the proton and anti-proton have different widths. This can be understood using the following argument: Let's say that each bunch can be described as a Gaussian with some RMS, then each bunch propagation in the Z direction can be expressed as

$$F_p \sim exp[\frac{-(z-ct)^2}{2\sigma_p^2}],$$
$$F_{\bar{p}} \sim exp[\frac{-(z+ct)^2}{2\sigma_{\bar{p}}^2}],$$

where σ_p^2 and $\sigma_{\bar{p}}^2$ are the widths of proton and anti-proton bunches respectively. The collision PDF is therefore:

$$F_{p-\bar{p}} \sim exp(-(z-ct)^2/(2\sigma_p^2) - (z+ct)^2/(2\sigma_{\bar{p}^2})).$$

To find the average collision time, we set $dF_{p-\bar{p}}/dt = 0$ and after some algebra find:

$$ct = z\frac{\sigma_{\bar{p}} - \sigma_p}{\sigma_{\bar{p}} + \sigma_p}.$$

One can estimate the RMS of the vertex Z position as $RMS_Z = \sigma_p * \sigma_{\bar{p}}/\sqrt{\sigma_{\bar{p}}^2 + \sigma_p^2}$ and calculate from this and the above equations the width of the proton and anti-proton bunches. This is illustrated in Fig. 15. If we take the RMS_Z from Figure 5, after some algebra one can get

$$\sigma_p \approx 55 \text{ cm and } \sigma_{\bar{p}} \approx 65 \text{ cm}.$$

5 Conclusion

We have presented a new space-time vertexing algorithm using COT tracks. Our performance studies indicate that the method is very efficient for events with 4 or more tracks and has excellent Z and timing resolution. There is also strong evidence that it does an much better job of separating collisions than the standard ZVertex algorithm. For those cases where there is reason to believe that two collisions have been combined, there is no evidence that there is a bias in the timing measurement or any degradation in the timing resolution measurement. While it was designed for use in the search for heavy, long-lived objects, we have every reason to believe this algorithm should work well for other searches and measurements.

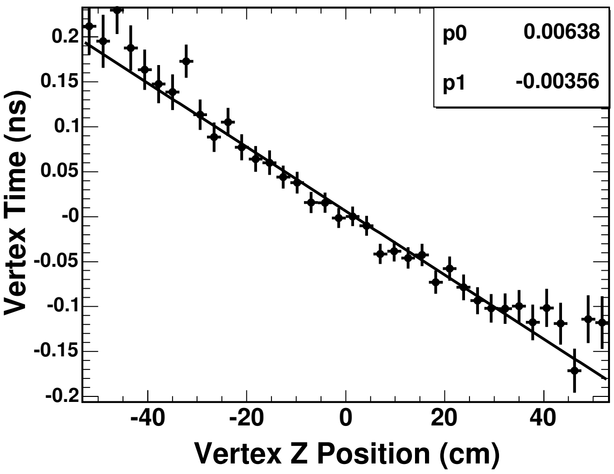

Figure 15: A profile histogram showing the mean vertex time as a function of the Z position of the collision. This is a real effect related to the difference in width between the proton and anti-proton bunches.

6 Appendix A

Let's denote q to be a fraction of a min-bias event to be a triggered event. Let min-bias events to be Poisson distributed with mean r, then the probability to observe n min-bias events is

$$r = L * \sigma / f$$
$$P_n = e^{-r} r^n / n!$$

where L is instantaneous luminosity, sigma is the cross-section (30 mb), and f is the average frequency of collisions.

In each of these n events the probability to observe m triggered events is given by binomial probability

$$q^m (1-q)^{n-m}.$$

The number of interactions in the triggered events is given by

$$< n >= 1/N * \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} n C_n^m P_n q^m (1-q)^{n-m} = r(1 - e^{-rq}(1-q))/(1 - e^{-rq}),$$

where the normalization is given by

$$N = \sum_{n=1}^{\infty} \sum_{m=1}^{\infty} C_n^m P_n q^m (1-q)^{n-m} = 1 - e^{-rq}.$$

In the limit of very small probability q one gets for the number of interactions in any triggered event:

$$< n >_{\lim_{q \to 0}} = 1 + r - rq. \tag{4}$$

One can see that the number of interactions is linearly proportional to the luminosity, and should be equal to one when luminosity is small.

References

[1] B.Cooper, M.D'Onofrio *et al.*, "Z-Vertex Finder", CDF Note 7227.

[2] W. Orejudos, R. Snider *et al.*, "Run 2 search for charged massive stable particles", CDF Note 6307.

[3] M. Goncharov, V. Krutelyov, D. Toback and P. Wagner, "Search for Neutral, Long-Lived Particles that Decay to Photons", CDF Note 8016.

[4] T. Phillips, "Event T_0 from the COT and the CLC", CDF Note 6277.

[5] McLachlan, G. and Peel, D., "Finite Mixture Models," Wiley-Interscience (2000).

[6] MacQueen, B., "Some Methods for classification and Analysis of Multivariate Observations," Proceedings of 5$^{\text{th}}$ Berkeley Symposium on Mathematical Statistics and Probability, Berkeley, University of California Press, 1:281-297 (1967).