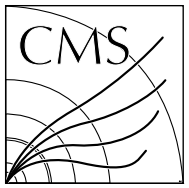
Available on CMS information server

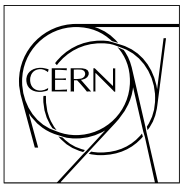
CMS NOTE-2008/033



The Compact Muon Solenoid Experiment

CMS Note

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland



02 July 2008

Adaptive Vertex Reconstruction

Wolfgang Waltenberger

Abstract

A very simple but powerful general-purpose vertex finding algorithm is presented that is based on the well-established adaptive vertex fitter to find and fit primary and secondary vertices.

1 Introduction

Reconstruction of interaction vertices is an essential step in the reconstruction chain of a modern collider experiment such as CMS; the primary ("collision") vertex is reconstructed in every event within the CMS reconstruction program, CMSSW. However, the task of finding and fitting secondary ("decay") vertices also plays an important role in several physics cases such as the reconstruction of long-lived particles like K_s, or the identification of b-jets, i.e. the task of b-tagging [1].

"Vertex finding" is defined as the task of sorting a set of tracks into subsets that share a common point of origin. Vertex fitting in this nomenclature refers to the task of computing the location and the error (i.e. covariance matrix) of an interaction vertex from a given set of reconstructed tracks.

Vertex fitters can be made robust i.e. insensitive to outlying or mis-measured tracks. Several robust vertex fitters have been proposed previously in CMSSW [2], one of which – the adaptive vertex fitter (AVF) – has been shown to be a powerful general purpose algorithm [3, 4, 5].

This publication presents a vertex finder method which harnesses the power of the AVF. It is an exceptionally simple method: the AVF is used iteratively to find and adaptively fit vertices, one by one. It is capable of finding and fitting both primary and secondary vertices. The algorithm is called the "adaptive vertex reconstructor" (AVR).

The paper is structured as follows. Sec. 2 lists the software and event topologies used for this paper. Sec. 3 describes all algorithmic aspects of the study. Sec. 4 tunes the algorithm to the application of b-tagging. Finally, Sec. 5 concludes and summarizes the study.

2 Event samples

This study was performed with CMSSW_2_0_12. Pythia is the event generator of all events used in this paper. No mis-alignment and ideal detector conditions are assumed. The following samples are used in this publication:

- 10000 $b\bar{b}$ di-jet events ($50 < \hat{p}_T < 120$ GeV, MSEL=0),
- 10000 $c\bar{c}$ di-jet events ($50 < \hat{p}_T < 120$ GeV, MSEL=0),
- 130000 "low energy" QCD events ($50 < \hat{p}_T < 120$ GeV, MSEL=1),
- 25000 "energetic" QCD events ($120 < \hat{p}_T < 170$ GeV, MSEL=1),
- 10000 $t\bar{t}$ events ($50 < \hat{p}_T < 120$ GeV, MSEL=0).

Fig. 1 shows the Monte Carlo truth of the B-Meson flight paths in the $b\bar{b}$ and $t\bar{t}$ event samples.

3 Algorithms

This section presents the algorithms used in this paper. Starting with a recap of the AVF method (Sec. 3.1), a "classical" algorithm will be presented which will serve as the baseline algorithm to compare with (Sec. 3.2). Finally, the novel AVR method will be presented (Sec. 3.3). In order to obtain a physically meaningful performance comparison, a simple flavour tagger will be employed (Sec. 3.4).

3.1 Adaptive Vertex Fitter

The AVF [3, 4, 5] is an iterative weighted Kalman filter [6]. Each track is given a weight which is a function of the reduced distance between the track and the vertex candidate. If χ_i^2 denotes the square of the standardized residual, then the weight w_i equals:

$$w_i\left(\chi_i^2\right) = \frac{\exp(-\chi_i^2/2T)}{\exp(-\chi_i^2/2T) + \exp(-\sigma_{\text{cut}}^2/2T)}. \tag{1}$$

A cutoff parameter σ_{cut}^2 has been introduced which is defined as the standardized residual for which $w_i = 0.5$, and a "temperature" parameter T, which defines "softness" of the weight function. This weight can be interpreted as a track-to-vertex assignment probability. Instead of minimizing the least sum of squares, the algorithm now

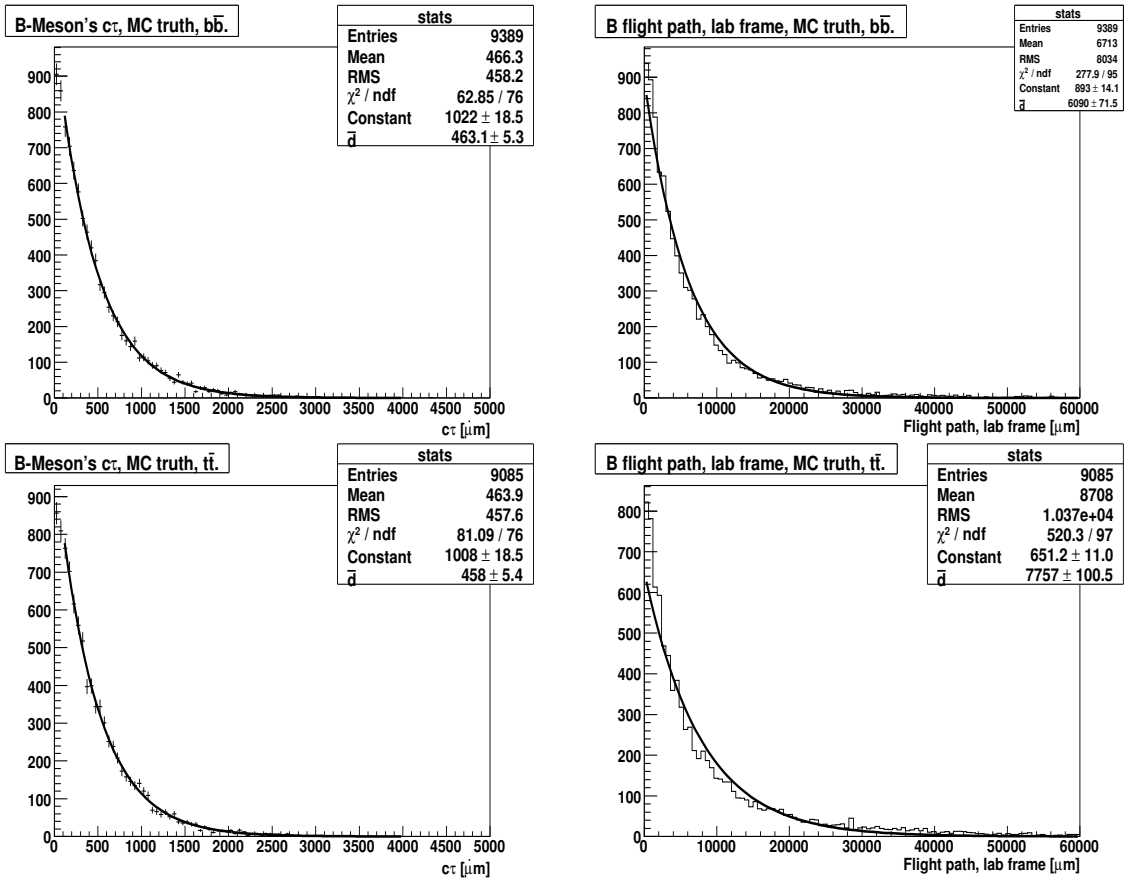


Figure 1: Monte Carlo truths of B-Meson's $c\tau$ (left) and flight path in the lab frame (right), $b\bar{b}$ sample (top) and $t\bar{t}$ sample (bottom). Exponential functions have been fitted into the histograms; the parameter labeled $\bar{d}$ in the stats box refers to the fitted flight path.

minimizes the *weighted* least sum of squares. In order to avoid falling prematurely into local minima, a deterministic annealing schedule is introduced: in each iteration step the temperature parameter T is lowered. Typically, a "quasi-geometric" annealing schedule that converges towards 1 is employed:

$$T_i = 1 + r \cdot (T_{i-1} - 1) \tag{2}$$

Here, T_i refers to the temperature parameter T at iteration i. r denotes the annealing ratio. For convergence, $0 < r < 1$ is needed. Fig. 2 shows the weight as function of χ_i^2 and T.

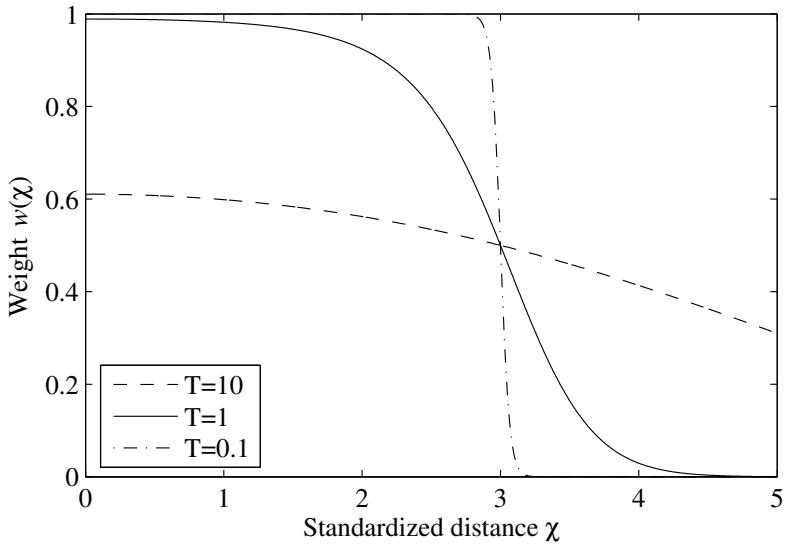


Figure 2: The weights of the AVF, as a function of the standardized residual $\chi \equiv \sqrt{\chi^2}$ and the annealing temperature T. For this plot, σ_{cut} has been chosen to be 3.0. This corresponds to ≈ 1 % of all inlier track being assigned a weight of less than 0.5, in the case of ideal data.

3.2 Baseline algorithm

The TrimmedKalmanVertexFinder (TKVF, see [5]) will serve as a baseline for the performance studies. This study will *not* systematically compare the AVR with this algorithm, as the competition would be unfair. The TKVF algorithm is used with default settings; no tuning has been performed.

The TKVF is a "classical" method that iteratively uses a Kalman vertex fitter. In each iteration step the track that is least compatible with the current vertex candidate is discarded. The vertex fit is repeated until no incompatible tracks remain. The procedure is then applied to the remaining set of incompatible tracks.

3.3 Adaptive Vertex Reconstructor

Given the AVF, the adaptive vertex reconstructor (AVR) is straightforward and can be stated as a simple, recursive imperative:

Apply the adaptive method to the set of tracks that remain un-associated after the previous iteration.

This sentence conveys the basic idea but leaves out a few details. The following diagram defines the algorithm more explicitly:

Fit the primary vertex adaptively with all user-supplied tracks.

Optionally take into account the beamspot constraint, and "non-secondary tracks".

Consider all tracks except the "non-secondary" ones that
have been assigned a weight smaller than 0.5 in the last fit.

Call this set "remaining tracks".

Fit a vertex adaptively with the remaining tracks.

Repeat until no more vertices can be fitted.

Primary vertices differ greatly topologically from secondary vertices in CMS – e.g. their track multiplicities are an order of magnitude higher. This qualitative difference is taken into account in the AVR by treating its first iteration specially: a different cutoff parameter and a different annealing schedule can be defined for the first vertex fit (which virtually always reconstructs the primary vertex).

3.3.1 Fitting with a beamspot constraint

The region in which the proton-proton collisions occur is called the *interaction region* or the *beamspot position* of the experiment [7]. The knowledge of this beamspot position can be used as *prior information* to improve vertex fitting, and, consequently, vertex finding.

The Monte Carlo samples used were produced with a beam width of 31.7 μm (measured in the transverse plane) and a beam length of 5.3 cm. Fig. 3 shows the impact of the beamspot constraint on the final (primary) vertex fit for the AVR, $b\bar{b}$ sample. It can clearly be seen that the x and y resolutions benefit from this additional information, while the z coordinate exhibits only a marginally better resolution. Also, the distributions tend to loose their tails – they are closer to an ideal Gaussian shape. These results are compared with the baseline (TKVF), in Fig. 4. The ability to deal with beamspot information has been added to TKVF specifically for this note.

3.3.2 Track refitting

The information of the vertex position can be used to improve the track fit by constraining the track to the vertex. In the Kalman filter such a constraint emerges naturally – the estimate of the track parameters are a result of the minimization procedure, as described in detail in [6]. This procedure is called "track refitting", or "track smoothing". In the "weighted" formulation of the Kalman formalism, the track weights have to be taken into account. This is simply done by replacing all occurrences of the tracks' covariance matrices $\mathbf{C_i}$ by the "weighted" covariance matrices $w_i^{-1}\mathbf{C_i}$.

Fig. 5 shows a contour plot of the residuals of the two "positional" track parameter: "transverse impact parameter" (d_0) and "longitudinal impact parameter" (z_0). The influence of track smoothing on the track parameters is clearly visible. The beamspot constraint improves the reconstructed primary vertex and thus further narrows down the track parameter resolutions. Fig. 5 also documents the additional improvement of the beamspot constraint on the track parameters. Fig. 6 shows the "RMS ellipsoids" of the track parameters. All plots have been produced with the $b\bar{b}$ sample. Tracks from the primary vertex as well as "displaced" tracks are included.

3.3.3 Adding tracks from other jets

Since the LHC is a hadron collider, many, if not most, tracks in CMS are part of a jet. For the purpose of finding secondary vertices within one specific jet only, it may be useful to supply the tracks from all other jets as a separate set of tracks. That way, an algorithm can search for the primary vertex in the sum of the two sets, while a secondary vertex candidate should contain tracks only from the first set. (Typically, the algorithm first removes the tracks associated with the primary vertex from this first set). This splitting of the set of tracks can thus further help the vertex finding algorithm to find and fit (interesting) secondary vertices more easily.

The AVR has been extended to account for this extra information. If not stated otherwise, in this paper, secondary vertex finding has been performed "per-jet", all other tracks being supplied as "non-secondaries", i.e. in the second

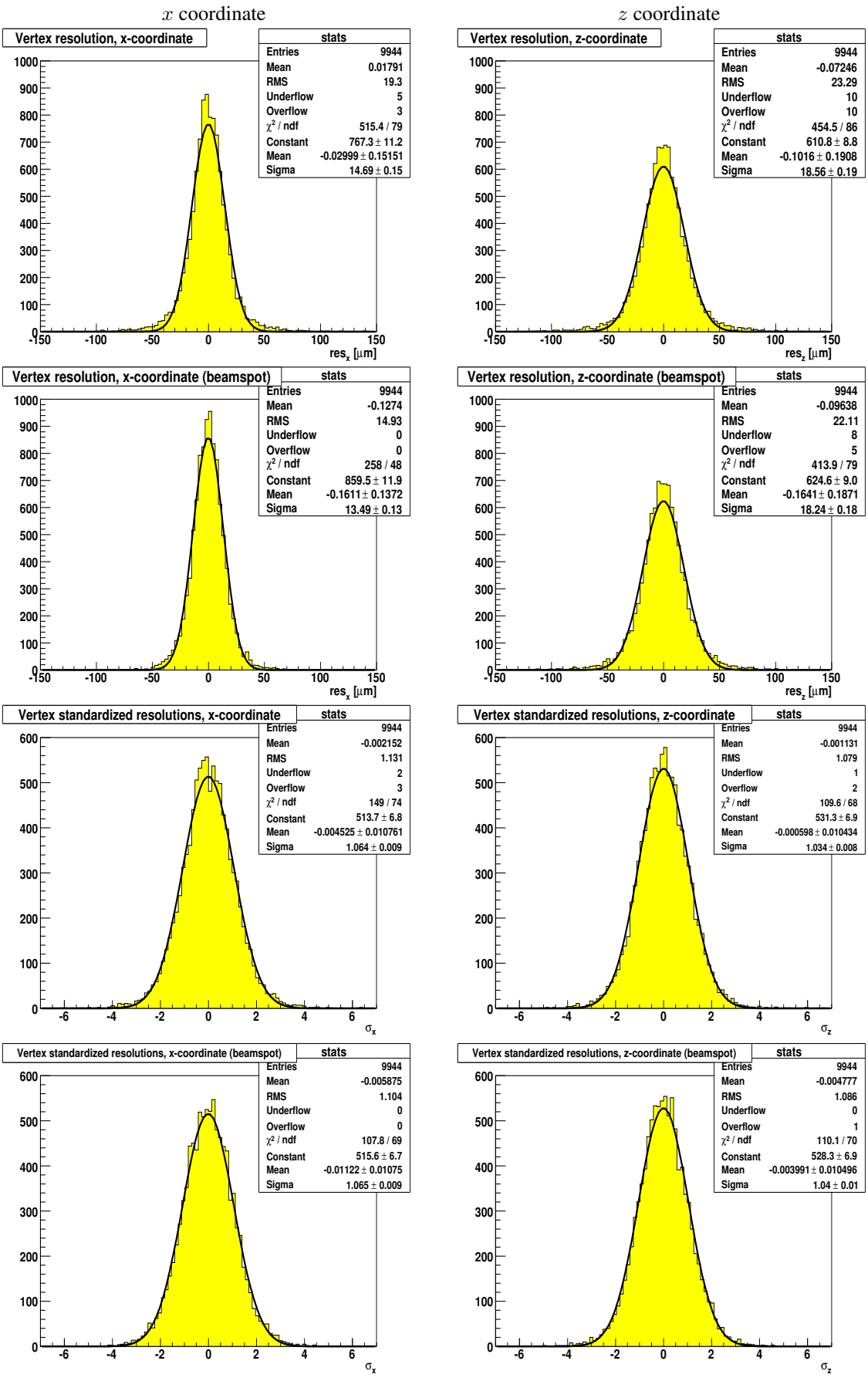


Figure 3: Vertex fit resolutions and standardized residuals, with and without beamspot constraint (AVR).

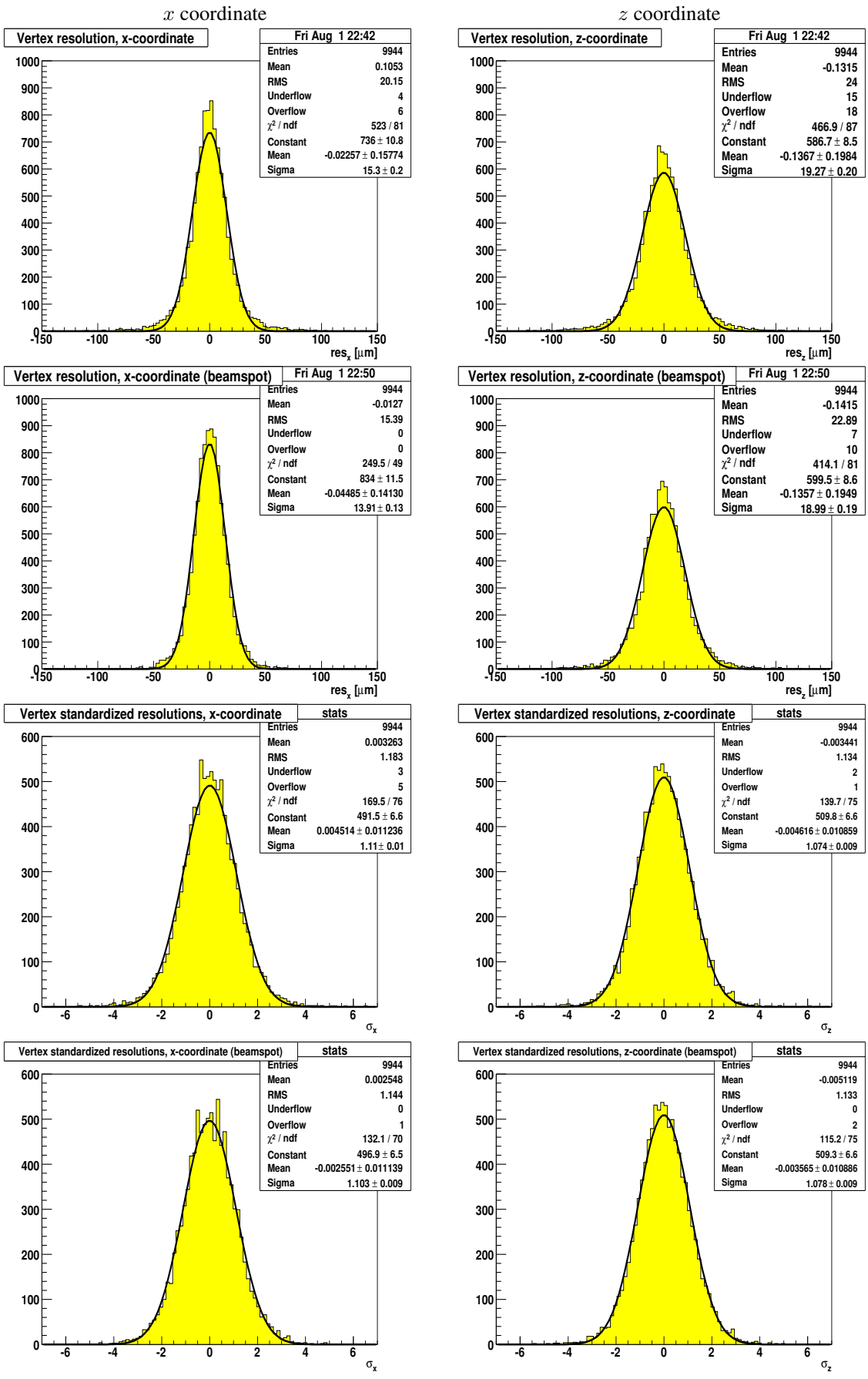


Figure 4: Vertex fit resolutions and standardized residuals, with and without beamspot constraint, for the baseline (TKVF).

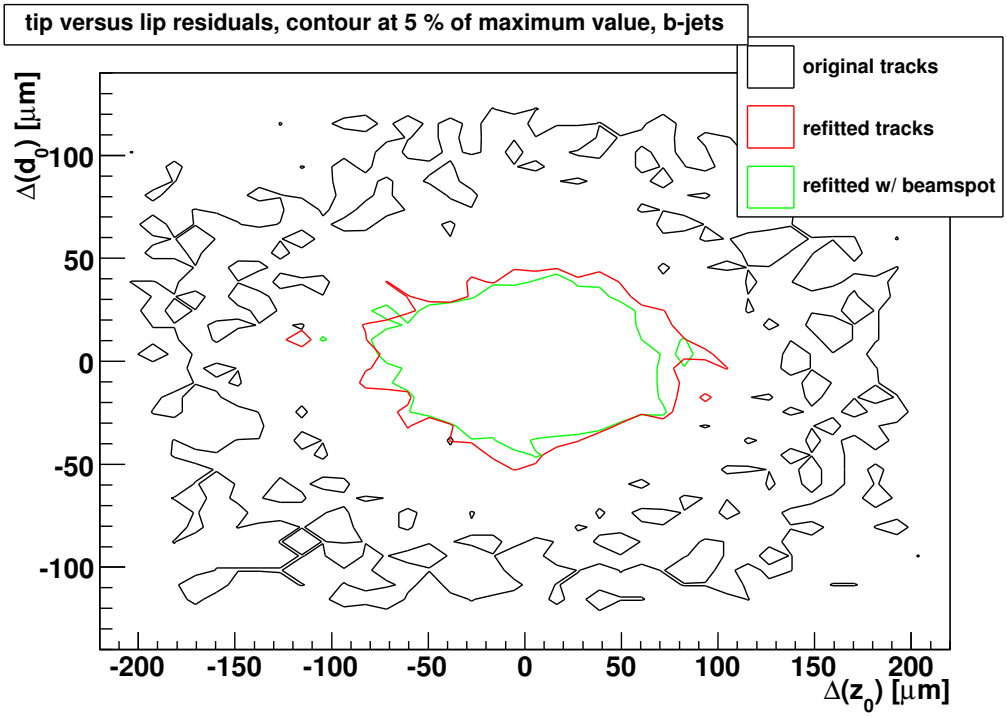


Figure 5: Contour plot of the tracks' d_0 and z_0 residuals. Track smoothing significantly reduces these track parameter resolutions. Fitting with a beamspot constraint adds to the gain.

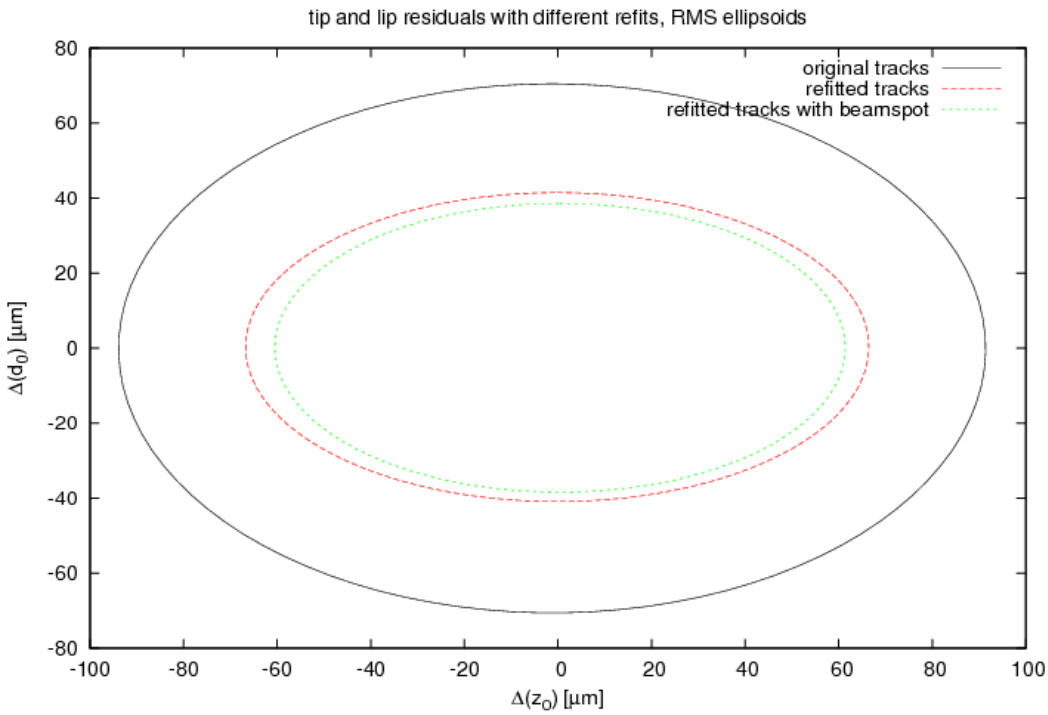


Figure 6: RMS ellipsoids of the tracks' d_0 and z_0 residuals.

set of tracks. Multiple interactions can result in more than one jet "belonging" to the same secondary vertex. Such effects are not accounted for in this publication.

3.3.4 Algorithm parameters

The user-definable parameters in the CMSSW implementation of the method are:

- $\sigma_{\text{cut,p}}$: the cutoff parameter for the first iteration (i.e. for the primary vertex),
- $T_{\text{ini,p}}$ and r_{p}: the annealing schedule for the first iteration,
- $\sigma_{\text{cut,s}}$, $T_{\text{ini,s}}$, and r_{s}: cutoff and annealing schedule for all subsequent iterations.

The CMSSW implementation has a few more configurable parameters:

- w_{min}: the minimum track weight below which a track is available in the next iteration in the iterative procedure. The default for this parameter is 0.5; from an algorithmic point of view there is no evident reason to change this parameter (albeit there may still be a physics case for a non-default setting).
- *smoothing* (boolean): Is track refitting ("track smoothing") applied to the tracks after vertex reconstruction?
- *weightthreshold*: if the weights of fewer than two tracks are below a certain threshold, the vertex is discarded. This is analogous to the situation in the AVF; it has been introduced to discard statistically problematic "one-prong" vertices [4] – the vertices' covariance matrices tend towards overestimation. In the context of b-tagging, overestimated vertex errors do not pose a problem. Therefore, *weightthreshold*=0 has been chosen for this particular use case.

3.4 Flavour tagger

CMSSW has a flavour tagger that exploits flight path information only: the "simple vertex-based tagger" [8]. This tagger employs a vertex reconstructor to find vertices. It then discards all vertices which are too close ($< 100\mu$m) to the primary vertex. Vertices which share too many tracks (65%) with the primary vertex are also discarded. From the remaining vertices one vertex is selected according to a certain, user-definable criterion. This vertex is then regarded as the "signal" secondary vertex.

The method's discriminator is a function of the distance between the primary and the "signal" secondary vertex. While other taggers show a better performance, this one depends most crucially on vertexing. Also, it is a tagger that needs no calibration; only a few parameters need to be defined. The tagging algorithm is run with default parameter settings which means:

- the discriminator is a function of the significance of the 3D flight path,
- the beamspot constraint is taken into account,
- if more than one secondary vertex is found, the one with the smallest uncertainty in the 3D flight path is chosen as the "signal" secondary vertex.

With s denoting the flight path significance, the discriminator value in the default setup is then equal to:

$$d = \ln(1 + s) \tag{3}$$

The vertex-based tagger operates on a per-jet basis. Tracks from other jets can thus be used to improve vertexing, as described in Sec. 3.3.3. For this analysis note the simple vertex-based tagger has been extended to account for this new vertexing feature.

4 Tuning to b-tagging

In order to achieve optimal b-tagging performance, the AVR must be tuned to this specific task. The tuning procedure has been split up into two different parts. In a first step, reconstruction of the primary vertex is optimised, reproducing some of the results given in a previous publication [4], with a more recent CMSSW software version. In a second step the cutoff parameter for secondary vertex fitting is tuned. Note that in reality, the procedure was applied iteratively. In all tuning steps the (future) default values for all parameters have been used, except for the one parameter that is under study in this specific step.

4.1 Choosing good parameter settings for the primary vertex

4.1.1 $\sigma_{\text{cut,p}}$

First a good $\sigma_{\text{cut,p}}$ needed to be found. This is the cutoff parameter for the primary vertex fit. Apart from determining the resolutions of the primary vertex, it also defines which tracks are associated to the primary vertex. These tracks will not be available for any subsequent fit. A too high value for $\sigma_{\text{cut,p}}$ should thus be avoided in the context of b-tagging.

A similar study has been performed before [4], but this time a much more recent setup of the CMS software and matching data has been used. Also, the case for fitting with a beamspot constraint is studied for the first time.

Figs. 7, 8 show the results of fitting a Gaussian distribution to the x and z residuals to extract the resolution. The histograms were cut off at $dz = 100\mu$m. The rms of the histograms is denoted as "rms", "σ" refers to the sigma of the fitted Gaussian distribution. "$> 100\mu$m" denotes the failure of finding a vertex within 100 μm from the Monte Carlo truth, independent of the actual reason.

$\sigma_{\text{cut,p}}$ has an essential influence on b-tagging behaviour. A $\sigma_{\text{cut,p}}$ parameter that optimises the primary vertex resolutions need not be the optimal value for b-tagging. Thus, as a cross check, a simple analysis to study the influence of $\sigma_{\text{cut,p}}$ was performed. Fig. 9 shows how the simple vertex-based tagger depends on this parameter. Fig. 10 shows the simple vertex-based tagger exploiting the new feature Sec. 3.3.3.

The default parameter setting for primary vertexing, $\sigma_{\text{cut,p}} = 3.0$ seems to be an adequate choice for b-tagging, also. The feature of adding tracks from other jets (Sec. 3.3.3) improves the b-tagging performance.

4.1.2 Annealing – $T_{\text{ini,p}}$

Subsequently, a good initial temperature of the annealing schedule needed to be found. While the centre of the distribution of the vertices' residuals should not be affected, the temperature could affect the number of failed fits. Also, CPU consumption depends critically on this parameter. Fig. 11 shows the dependence of the vertex fit on T_{ini}. The fit is very insensitive to this parameter, perhaps due to the high-multiplicity nature of the event samples. Only the CPU consumption is significantly affected by this parameter showing a positive, logarithmic correlation. Note that all CPU times given apply to the entire vertex reconstruction procedure – finding and fitting of primary and secondary vertices are included in these numbers, along with track re-linearization, if needed. It can be seen that the AVR is significantly faster than the baseline.

4.1.3 Annealing ratio r_{p}

Also, an annealing ratio r needed to be found. The annealing ratio should not affect the statistical properties of the vertex fit; a high ratio ($r \to 1$) could only stabilize the fit in the sense that fewer "failed" fits could be expected. A lower annealing ratio on the other hand should result in a faster algorithm.

Fig. 12 shows a scan in r. The results indicate that $r^{-1} \approx 2^7$ is a good choice, resulting in a fast algorithm, when compared with the baseline.

4.2 Reconstruction of B-Meson flight paths

Finding a good $\sigma_{\text{cut,s}}$ cutoff parameter is not as easy as was the case with $\sigma_{\text{cut,p}}$: it is not clear how "missed" vertices (i.e. simulated vertices which cannot be associated with a reconstructed vertex) and "fake" vertices (i.e. reconstructed vertices which cannot be associated with a simulated vertex) are to be taken into account. Therefore, a different approach for determining a good default value was taken: it shall be tried to find a good $\sigma_{\text{cut,s}}$ cutoff parameter setting for the AVR algorithm by reconstructing the flight paths of the B-Mesons. Secondary vertices are found and fit with the AVR method. The "B vertex" is identified as the reconstructed vertex with the smallest reduced distance (i.e. the distance divided by the error of the reconstructed vertex) from the true simulated vertex. In order to have a physically meaningful quantity, the flight path was divided by the relativistic γ factor, which has been taken from the Monte Carlo truth. Fig. 13 shows the resulting flight paths with their errors for various $\sigma_{\text{cut,s}}$ values (green curve), compared with the Monte Carlo truth (red strip). Here, an exponential curve plus a constant background have been fit into a certain interval ($[350\mu\text{m}, 4000\mu\text{m}]$) of the histogram of reconstructed flight paths. Fig. 14 shows these histograms explicitly for the baseline and the new default value ($\sigma_{\text{cut,s}} = 3.0$). Since this analysis may favor strategies which produce a plethora of secondary vertices, the total number of reconstructed vertices and the number of unassociated reconstructed vertices (Fig. 15) are supplied. It can be seen that the

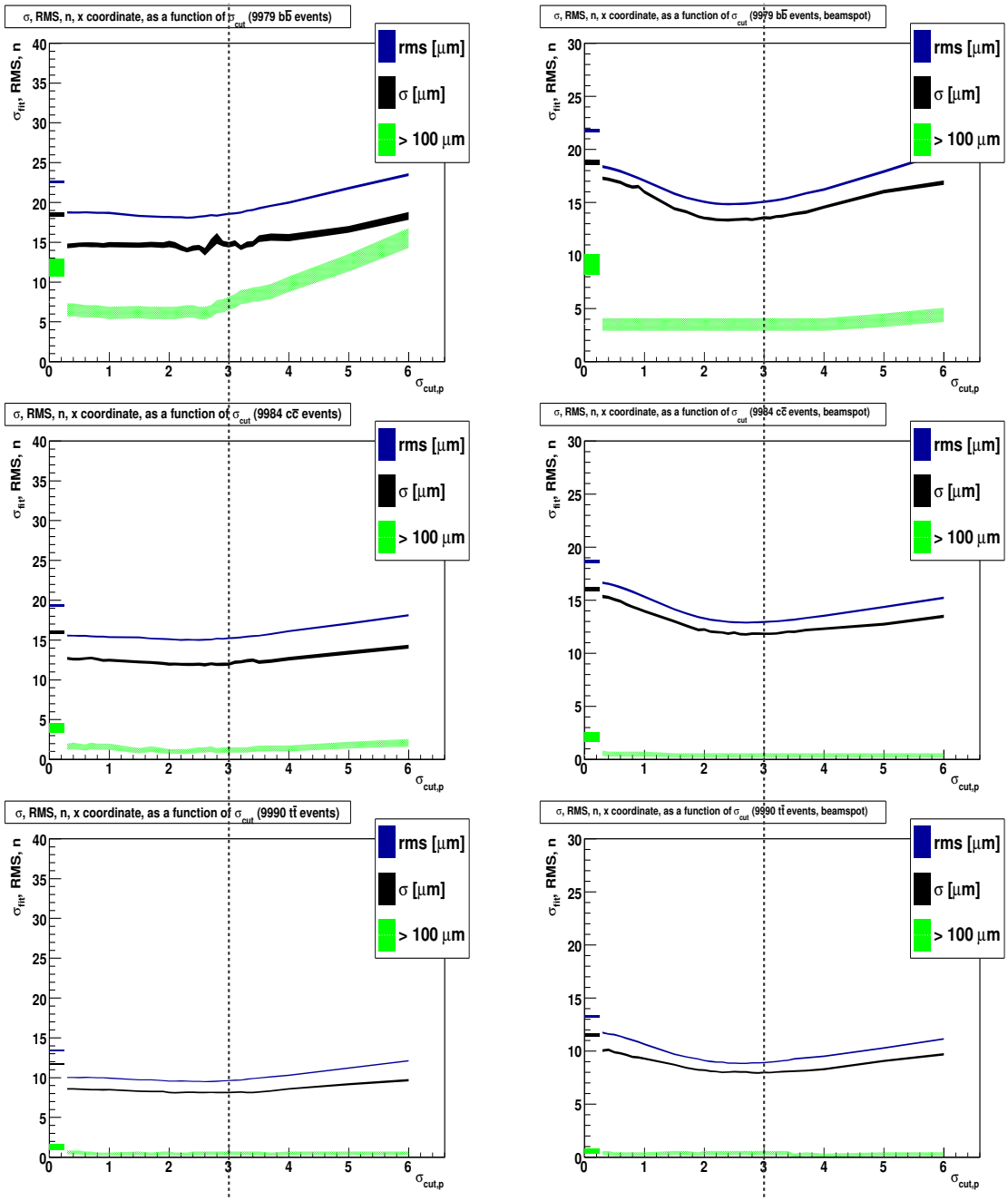


Figure 7: σ_{fit}, RMS, and fraction of "$> 100\mu$m" vertices, as a function of $\sigma_{\text{cut,p}}$, for the $b\bar{b}$ (top), $c\bar{c}$ (middle), and $t\bar{t}$ (bottom) events, without (left) and with (right) the beamspot constraint. $\sigma_{\text{cut,p}} = 0$ refers to the baseline algorithm (TKVF). Shown for the vertices' x coordinate. Note that "$> 100\mu$m" is given in $‰$.

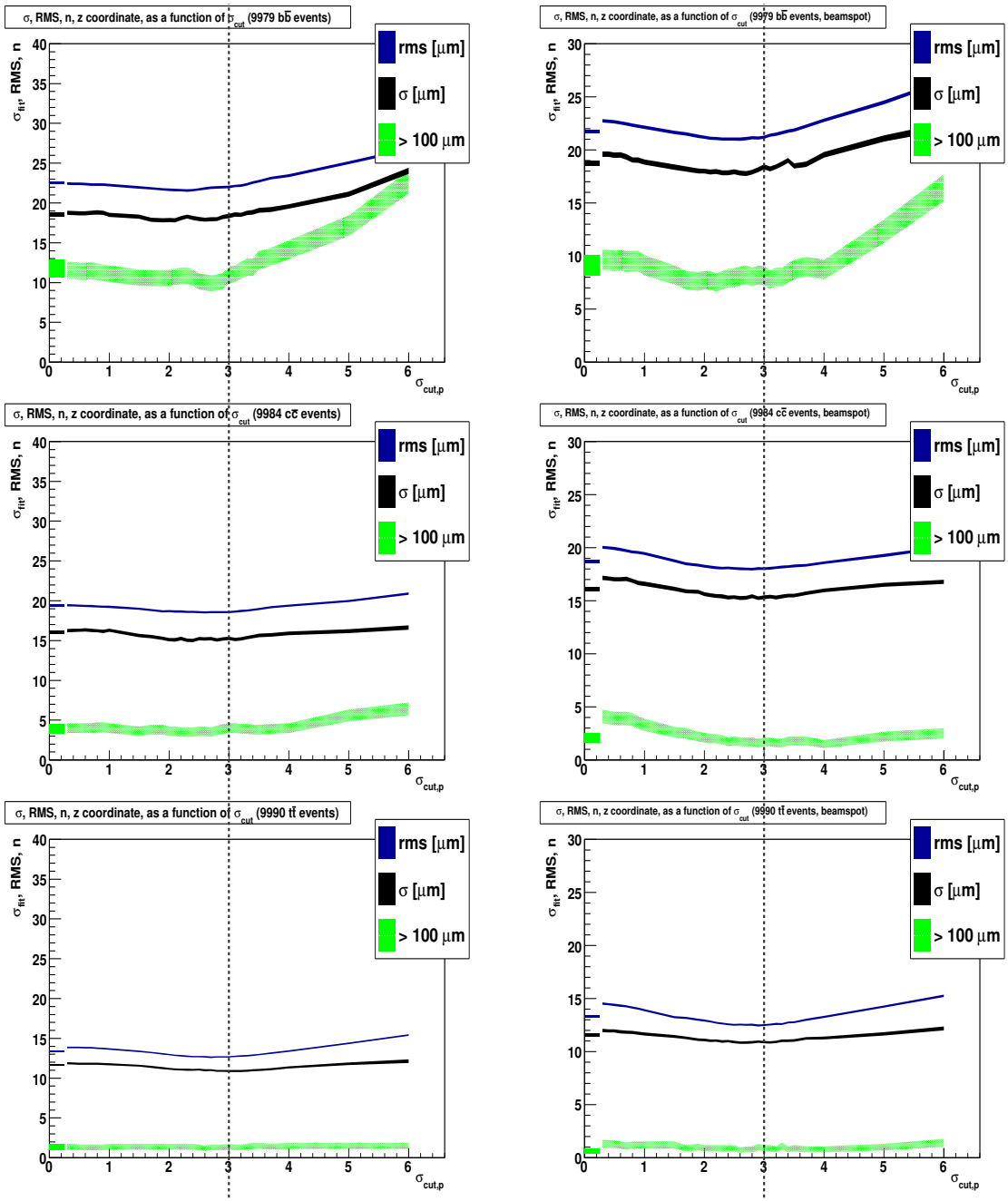


Figure 8: σ_{fit}, RMS, and fraction of "failed" fits, as a function of $\sigma_{\text{cut,p}}$, for the $b\bar{b}$ (top), $c\bar{c}$ (middle), and $t\bar{t}$ (bottom) events, without (left) and with (right) the beamspot constraint. $\sigma_{\text{cut,p}} = 0$ refers to the baseline (TKVF). Shown for the vertices' z coordinate.

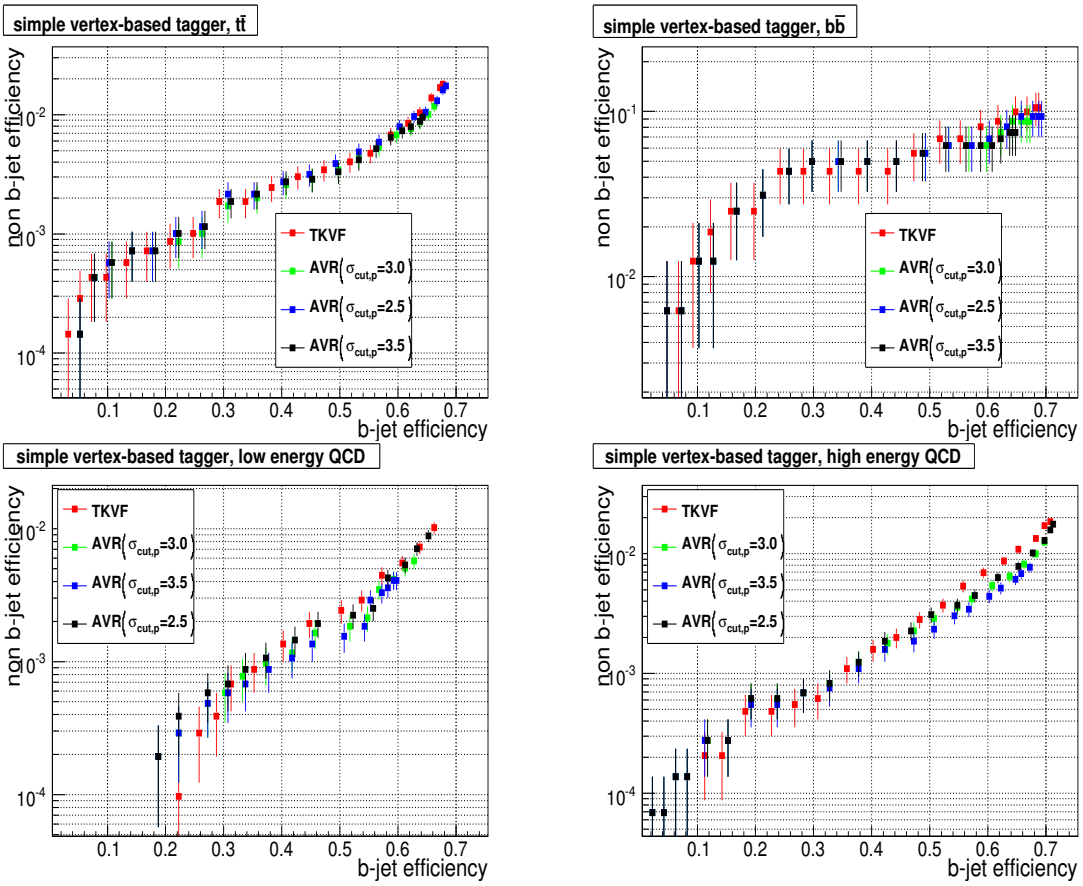


Figure 9: Performance of the simple vertex-based tagger, depending on $\sigma_{\text{cut,p}}$, for $t\bar{t}$ (top left), $b\bar{b}$ (top right), low energy QCD (bottom left), and high energy QCD (bottom right) event samples. Mis-tagging rates refer to light flavours, only. Gluons are excluded from the mis-tagging rate. Tracks from other jets have *not* been taken into account.

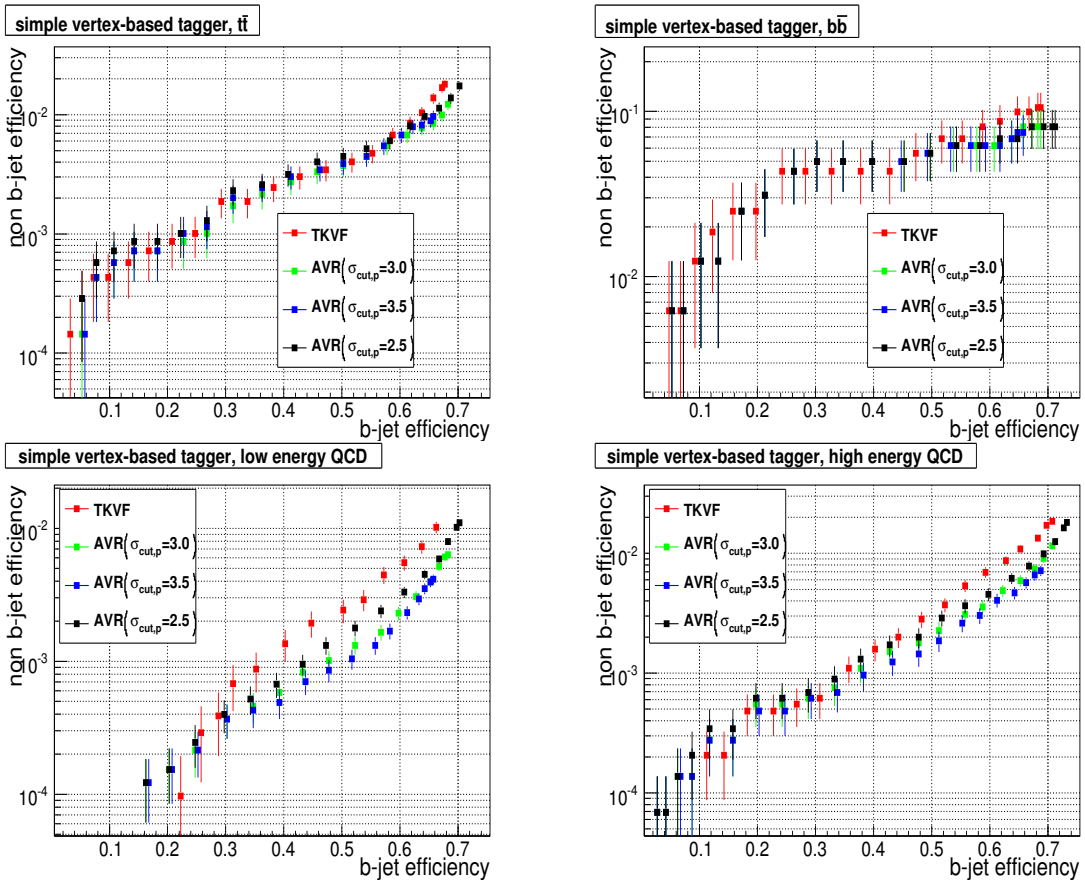


Figure 10: Performance of the simple vertex-based tagger, depending on $\sigma_{\mathrm{cut,p}}$, for $t\bar{t}$ (top left), $b\bar{b}$ (top right), low energy QCD (bottom left), and high energy QCD (bottom right) event samples. Mis-tagging rates refer to light flavours, only. Gluons are excluded from the mis-tagging rate. For the AVR, tracks from other jets have been taken into account as described in Sec. 3.3.3.

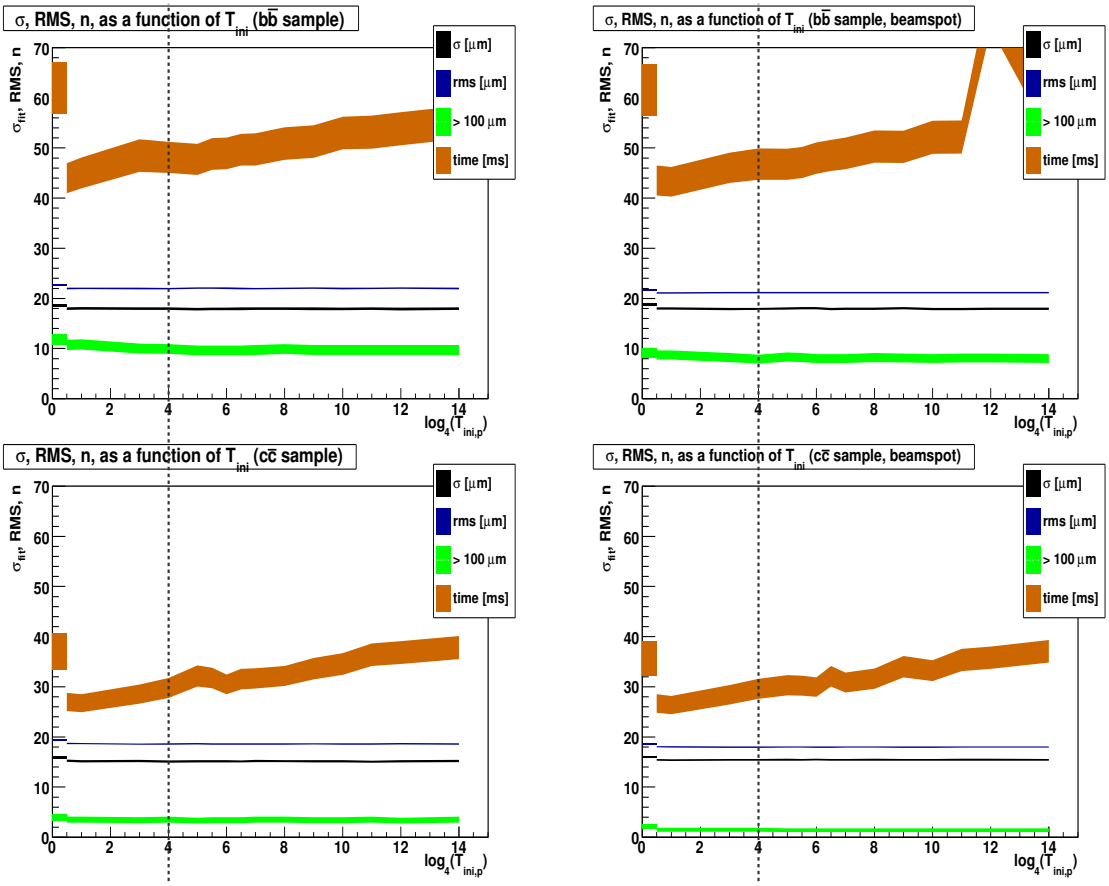


Figure 11: σ_{fit}, RMS, and fraction of "failed" fits, as a function of $T_{\text{ini,p}}$, for the $b\bar{b}$ (top) and the $c\bar{c}$ (bottom) events, without (left) and with (right) the beamspot constraint. $\log_4(T_{\text{ini,p}}) = 0$ refers to the baseline (TKVF).

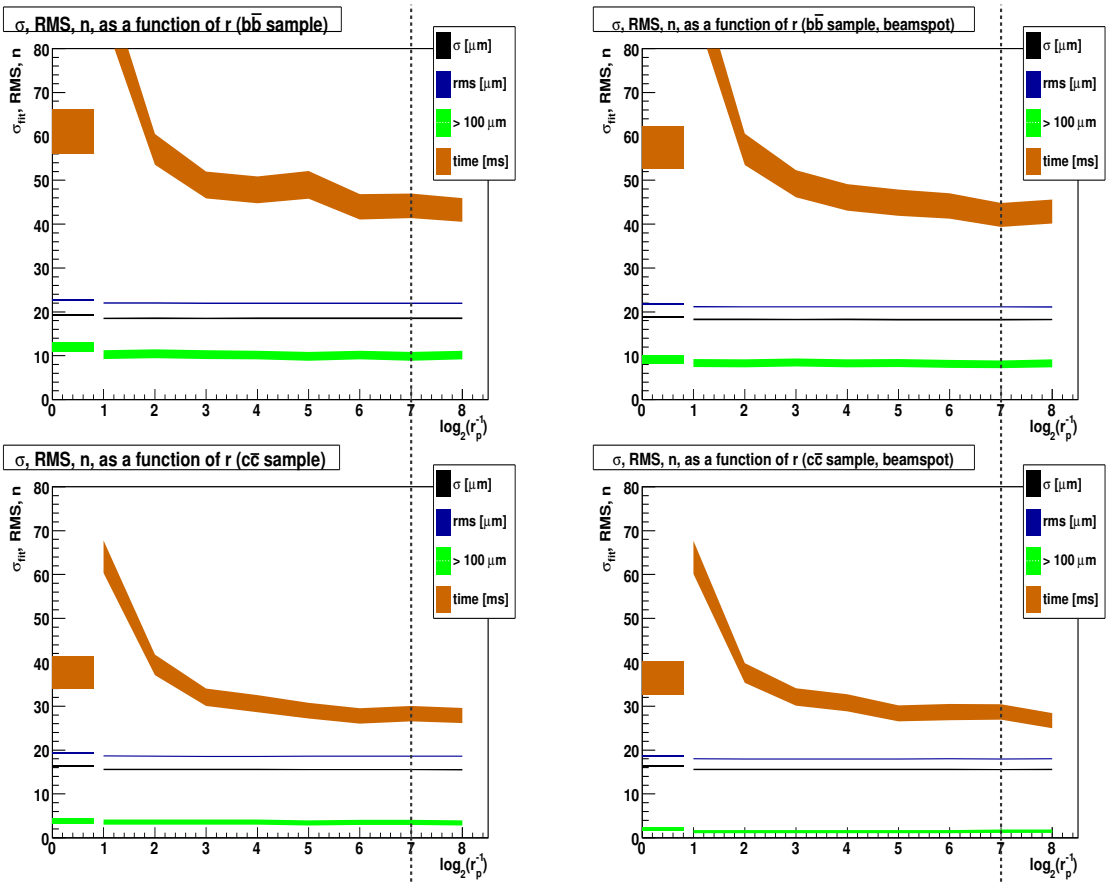


Figure 12: σ_{fit}, RMS, and fraction of "failed" fits, as a function of the annealing ratio r, for the $b\bar{b}$ (top) and $c\bar{c}$ (bottom) events, without (left) and with (right) the beamspot constraint. $\log_2(r_p) = 0$ refers to the baseline (TKVF).

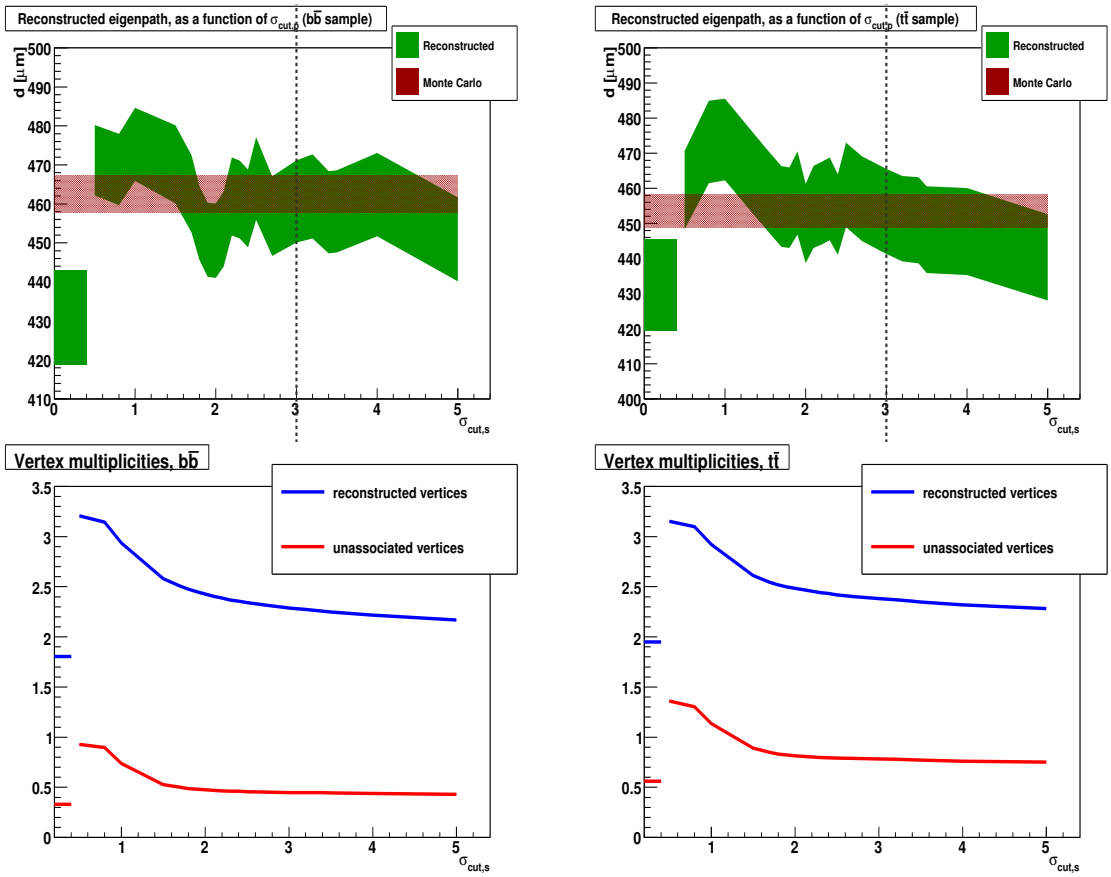


Figure 13: Reconstructed B-Meson's $c\tau$, for $b\bar{b}$ and $t\bar{t}$ (top), number of reconstructed vertices and un-associated vertices, in the same samples (bottom). $\sigma_{\text{cut,s}} = 0$ refers to the baseline (TKVF).

AVR algorithm quite generally tends to produce more vertices than the TKVF, both "good" vertices that can be associated with a simulated vertex, and fake vertices. The effect of this increased number of vertices on b-tagging is unclear and will be difficult to evaluate.

Note that algorithms that take tertiary vertices into account exist; they are not the subject of this study, though.

4.3 Vertex-based b-tagging

B-tagging performance was compared between the AVR and the baseline (TKVF) algorithms. The simple vertex-based tagger was taken as the b-tagging algorithm, since it is the tagger which most crucially depends on vertexing. In this setup the CMSSW defaults for this tagger are used. Only light quark flavours are included in the non b-jet efficiency. The jet flavour is identified by following the prescription detailed in [9], using the algorithmic definition, where the jet is matched to the heaviest parton within a cone of $\Delta R = 0.3$. This ensures that gluons which split into $b\bar{b}$ pairs are treated as b-jets. All matched jets with $p_t > 30\,\text{GeV}$ are used to measure the performance.

Fig. 16 shows a comparison between the TKVF and AVR. Fig. 17 shows a comparison between the TKVF and AVR, with the AVR exploiting the new feature (Sec. 3.3.3). Tab. 1 shows the CPU consumptions for the different event samples.

The AVR's b-tagging performs only slightly better than the TKVF. The use of the new feature of adding tracks from other jets Sec. 3.3.3 further boosts the performance albeit at the price of a much increased CPU consumption, see Tab. 1.

Filtering the primary tracks from the track set prior to b-tagging should result in the same b-tagging performance without the extra CPU cost. The author thus suggests that tracks with a "primary vertex assignment probability" above 0.5 should be discarded from the input set of tracks, prior to b-tagging. This strategy should result in a b-tagging performance that equals the performance of the new feature, without an increase in CPU consumption.

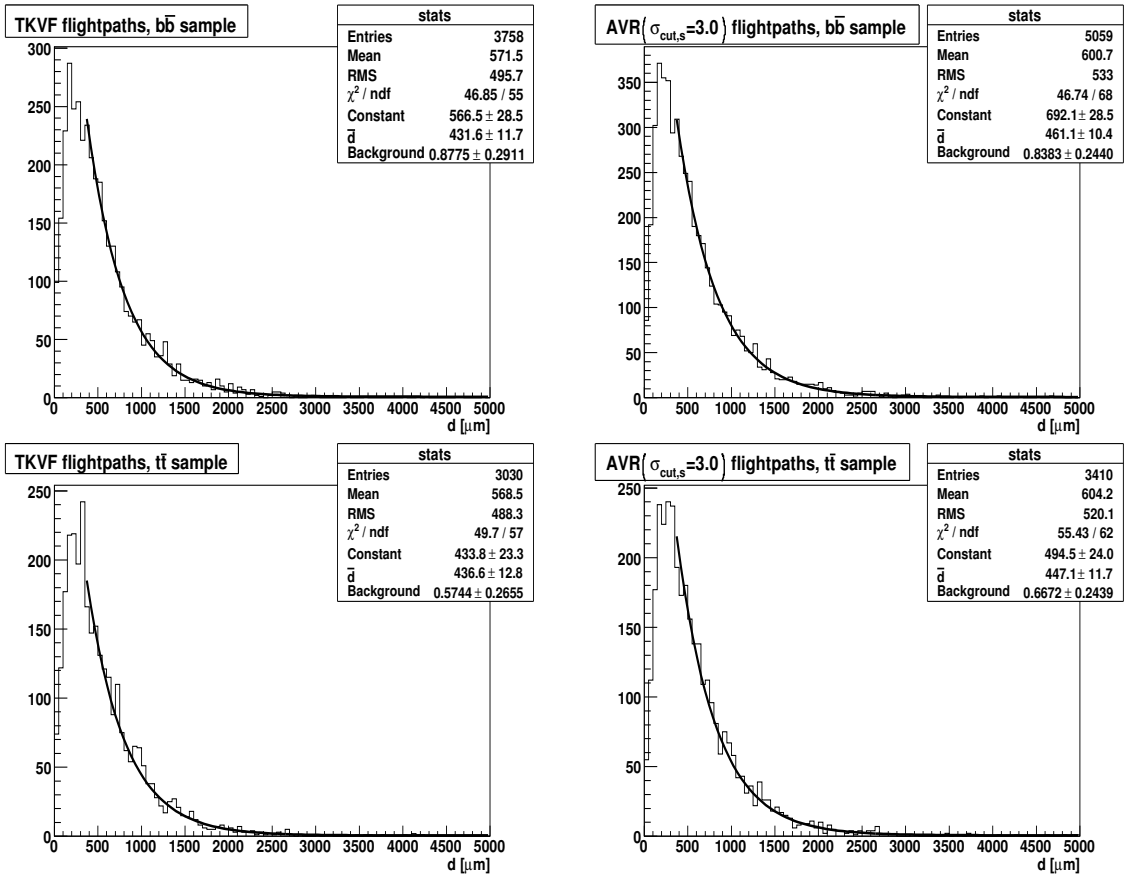


Figure 14: Flight path for TKVF(left) and AVR($\sigma_{\text{cut,s}} = 3.0$) (right), $b\bar{b}$ sample (top) and $t\bar{t}$ sample (bottom).

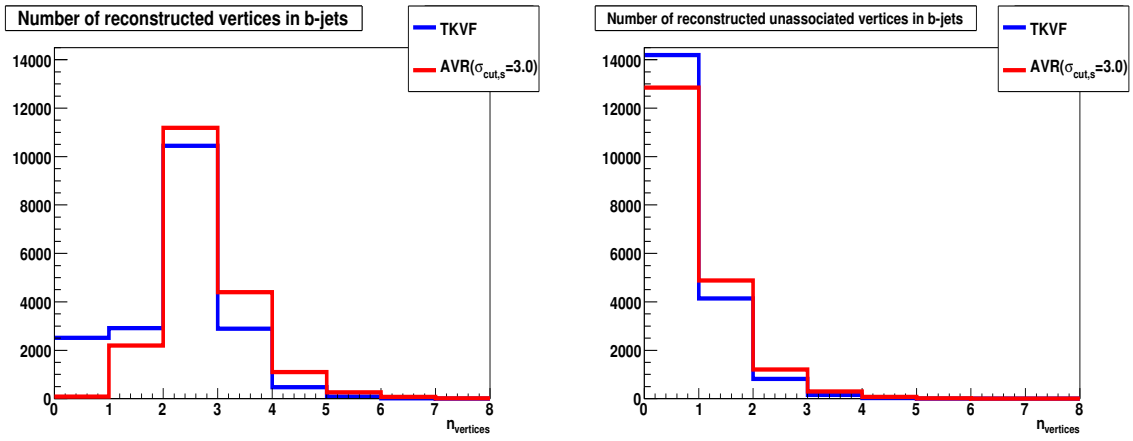


Figure 15: Total number of reconstructed vertices (left) and number of un-associated reconstructed vertices (right) in b-jet for TKVF (blue) and AVR($\sigma_{\text{cut,s}} = 3.0$) (red), $b\bar{b}$ sample.

event sample	AVR CPU [ms] one jet only	AVR CPU [ms] tracks from other jets
$b\bar{b}$	23	50
$t\bar{t}$	36	140
low energy qcd	24	90
high energy qcd	25	71

Table 1: CPU costs with (left) and without (right) exploiting the new feature (Sec. 3.3.3).

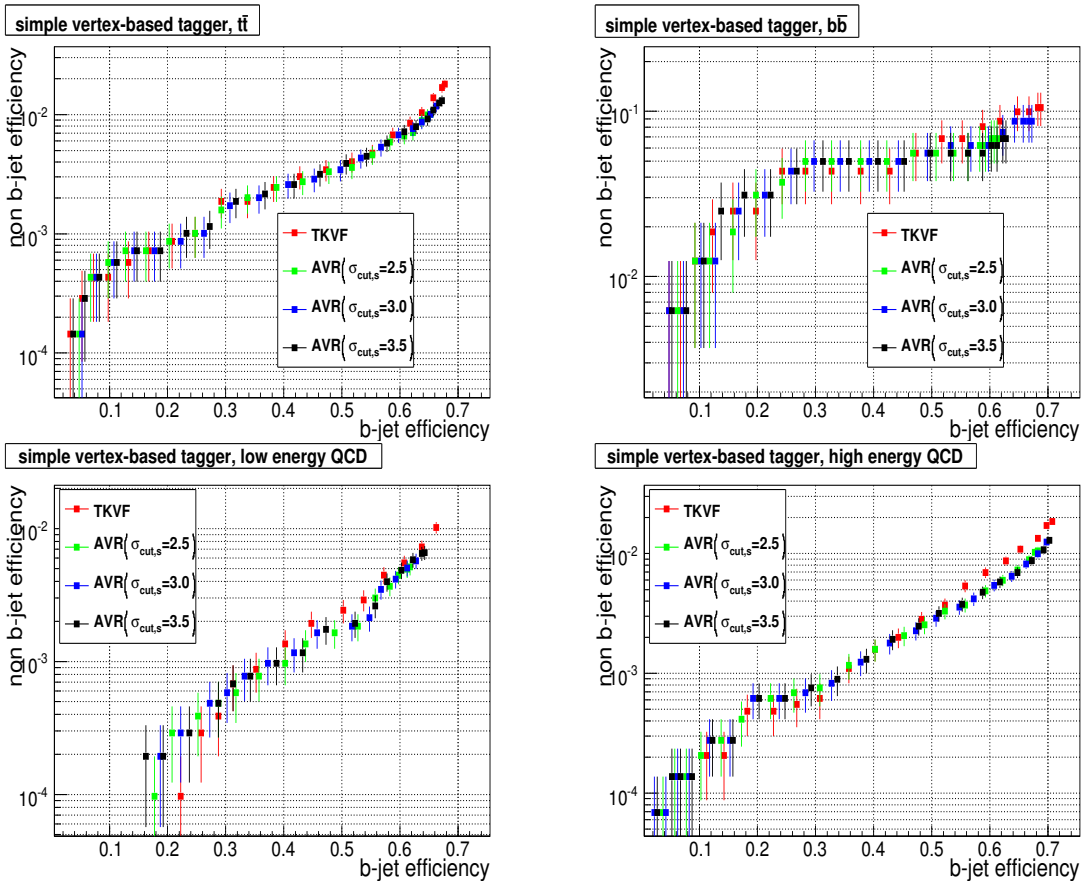


Figure 16: Performance of simple vertex-based tagger (algorithmic definition), depending on $\sigma_{\mathrm{cut,s}}$, for $t\bar{t}$ (top left), $b\bar{b}$ (top right), low energy QCD (bottom left), and high energy QCD (bottom right) event samples. Mis-tagging rate is for light flavours, gluons are excluded. Tracks from other jets have *not* been taken into account.

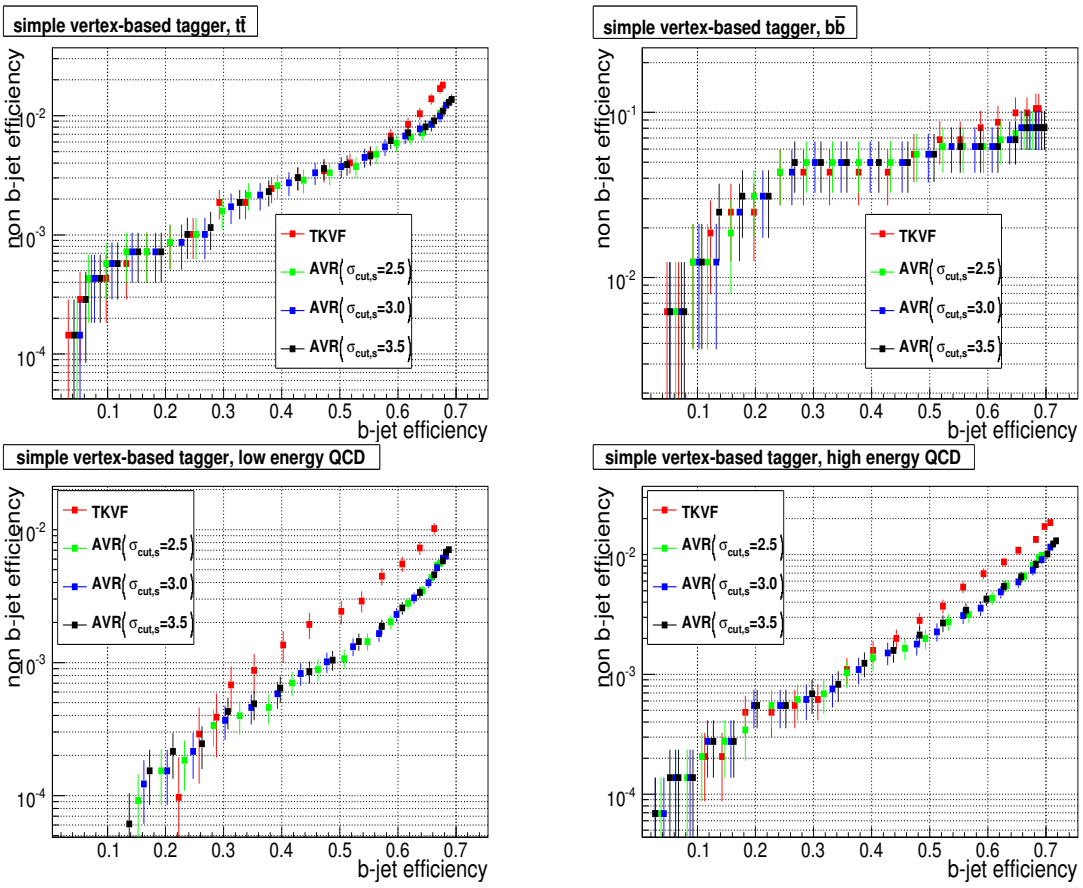


Figure 17: Performance of simple vertex-based tagger (algorithmic definition), depending on $\sigma_{\mathrm{cut,s}}$, for $t\bar{t}$ (top left), $b\bar{b}$ (top right), low energy QCD (bottom left), and high energy QCD (bottom right) event samples. Mis-tagging rate is for light flavours, gluons are excluded. For the AVR, tracks from other jets have been taken into account as described in Sec. 3.3.3.

5 Conclusion and Outlook

A simple but novel vertex finding algorithm, the adaptive vertex reconstructor, has been introduced and discussed. Default values have been found that cover a fairly wide range of sensible use cases. Compared with the baseline, the algorithm is faster and its overall performance is better, although it must be stressed that the baseline algorithm was taken "as-is": no optimization procedure was performed on this baseline algorithm.

It would be beneficial to study what can be achieved with an optimized version of the TKVF algorithm, e.g. in the context of b-tagging.

The feature of separating the jet tracks into two sets 3.3.3 has been shown to improve the performance of the simple vertex-based tagger, although at the price of a slower algorithm. A track-filtering step prior to b-tagging that discards tracks from the primary vertex should have the same effect on b-tagging, only without an increased CPU cost. This strategy should be tried in the near future.

When trained properly, the "combined vertex-based tagger" should outperform the simple vertex-based tagger. As a next step, it should thus be tried to optimize this algorithm for the combined method as well. This is a much more challenging task as the performance of the combined tagger depends critically on the calibration and millions of events are typically needed for the optimization process.

Acknowledgements

Thanks to Rudi Frühwirth for proofreading and valuable discussions. Thanks are also due to Thomas Speer, Tommaso Boccali, Kevin Stenson, Joanne Cole, and Fabrizio Palla for their highly appreciated comments.

References

[1] C. Weiser. A Combined Secondary Vertex Based B-Tagging Algorithm in CMS. (CMS NOTE-2006/014).

[2] W. Waltenberger. *Development of Vertex Finding and Vertex Fitting Algorithms for CMS*. PhD thesis, TU Wien, 2004. (CMS TS-2006/012).

[3] W. Waltenberger, R. Frühwirth, and P. Vanlaer. Adaptive Vertex Fitting. *Journal of Physics G: Nuclear and Particle Physics*, 34:N343–N356, 2007.

[4] R. Frühwirth, W. Waltenberger, and P. Vanlaer. Adaptive Vertex Fitting. (CMS NOTE-2007/008), 2007.

[5] T. Speer, K. Prokofiev, R. Frühwirth, W. Waltenberger, and P. Vanlaer. Vertex Fitting in the CMS Tracker. (CMS NOTE-2006/032), 2006.

[6] R. Frühwirth. Application of Kalman filtering to track and vertex fitting. *Nuclear Instruments and Methods in Physics Research A*, 262:444, 1987.

[7] T. Miao, H. Wenzel, F. Yumiceva, and N. Leioatts. Beam Position Determination using Tracks. (CMS NOTE-2007/021).

[8] CMS Collaboration. Impact of Tracker Misalignment on the CMS b-Tagging Performance. (CMS PAS BTV 07 003, 2008).

[9] CMS Collaboration. The CMS Physics Technical Design Report, Volume 1. *CERN/LHCC*, 2006-001, 2006.