DESY - ZEUS COLLABORATION

SUMMER STUDENT REPORT

Secondary vertex reconstruction from tracking data using the `ZEUS`-framework

$t \rightarrow$ 3-jet reconstruction from ZEUS-data at $\sqrt{s} = 318\,\mathrm{GeV}$

Author:
Harald VIEMANN

Supervisors:
Achim GEISER
Andrii GIZHKO

September 12, 2014

Abstract

The first part of the report discusses the usage of the `ZEUS-title-library` to reconstruct jets and secondary vertices from tracking-data taken by the ZEUS-detector and to port the algorithm for the usage on CMS-data. Further, in the second part, the production of a t quark at the ZEUS experiment has been investigated by a basic analysis of a $t \rightarrow$ 3-jet decay.

Contents

1. Secondary vertex reconstruction from tracking data using the `ZEUS`-framework

1.1 Introduction

The usage of the `ZEUS-title-library` offers the ability to reconstruct jets and secondary vertices from tracks without the usage of calorimeter information. A first try to adapt the ZEUS vertex reconstruction has been done in the summer-student-project 2013 [1]. Information about the ZEUS- or CMS-detector characteristics can be found in the report [1] of the year 2013.
The following discusses bug-checking and improvement of the original code. This report refers to code in the file '`TMiniNtupleAnalyzer.cxx`'.

1.2 CMS to ZEUS parametrisation

To be able to reconstruct jets and vertices from CMS-data with the ZEUS-library the track-parameter have to be transformed from CMS to ZEUS-parametrisation. A track gets described via a helix and the error via a track-specific covariance matrix. The following shows the transformation for the helix-parameter and the covariance matrix.

1.2.1 Helix-Parameters

1.2.1.1 The ZEUS- and CMS-Helix-Parameter

The ZEUS- and CMS-analysis-frameworks both use curvilinear coordinates to describe the track-helix [2][3]. Both parametrisations consist of 5 parameters and are stored into an array. Tabular 1.1 lists the specific helix-parameters as well as their ID for accessing the array. Further the analogous ZEUS-parameter for CMS-parameter are listet and their physical meaning gets mentioned.

1.2.1.2 Transformation of Helix-Parameter

The transformation of all CMS-parameter to ZEUS-parameter works directly except the transformation to $z_1 = \frac{Q}{R}$.

ZEUS			CMS			
ID	Name	Parameter	ID	Name	Parameter	$\hat{=}$
0	z_0	Φ	0	c_0	$\frac{q}{\lvert p\rvert}$	
1	z_1	$\frac{Q}{R}$	1	c_1	λ	Θ
2	z_2	QD_0	2	c_2	Φ	Φ
3	z_3	z	3	c_3	dxy	D_0
4	z_4	$\cot\Theta$	4	c_4	dsz	z

Table 1.1: Helix-parameter for ZEUS and CMS. Analogous ZEUS-variables are named for CMS-parameter. Φ describes the polar-angle of the helix, Θ states the tilt referring to the B-field, D_0 is the offset and z the z-shift of the reference point. R is used for the radius, p for the track-momentum and Q, q both tell the charge.

Transformation to z_1:

For the transformation to z_1 we remember the calculation of the helix-radius

$$R = \frac{mv_\perp}{QB} = \frac{p_\perp}{QB} \tag{1.1}$$

via Lorentz and centrifugal force. Because of we state that the magnetic flux density should be homogenous and constant around the vertex-regions the radius R is proportional to $p_\perp$ and z_1 has a direct link to c_0.

With these informations it is easy to transform c_0 to z_1:

$$z_1 = \frac{Q}{R} = B \cdot \frac{Q^2}{p_\perp} \hat{=} B \cdot \frac{q^2}{p_\perp} \tag{1.2}$$

with $p_\perp = p \cdot \sin\Theta$

$$= B \cdot \frac{q^2}{p \cdot \sin\Theta} = B \cdot c_0^2 \frac{p}{\sin\Theta} \tag{1.3}$$

Transformation of all parameter:

Now we can easily write down the transformations of all CMS- to ZEUS-parameter as shown in tabular 1.2.

ZEUS	CMS to ZEUS
z_0	c_2
z_1	$c_0^2 \cdot \frac{p}{\sin\Theta}$
z_2	$c_3 \cdot c_0 \cdot p$
z_3	c_4
z_4	$\cot c_1$

Table 1.2: Transformation of the CMS- to the ZEUS-helix-parameter.

1.2.2 Covariance-Matrix

After the transformation of the helix parameter we have a new covariance matrix. To get this new matrix we have to transform the CMS- to the ZEUS-covariance-matrix.

Jacobian:

The parameter-transformation is a nonlinear transformation of the form $\vec{z} = \varphi(\vec{c})$. Since there is no exact formula to describe the transformation of the covariance matrix we take a look at the taylor expansion of the transformation

$$\vec{z} = \varphi(\langle\vec{c}\rangle) + J(\langle\vec{c}\rangle) \cdot (\vec{c} - \langle\vec{c}\rangle) + \mathcal{O} \cdot (\vec{c} - \langle\vec{c}\rangle)^2 \tag{1.4}$$

with the Jacobian matrix

$$J(\langle\vec{c}\rangle) = \frac{\partial(z_0, z_1, z_2, z_3, z_4)}{\partial(c_0, c_1, c_2, c_3, c_4)} \quad . \tag{1.5}$$

With the transformations shown in tabular 1.2 we get for the Jacobian matrix

$$J(\langle\vec{c}\rangle) = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 \\ 2p \cdot B \cdot c_0 \cdot \csc c_1 & -p \cdot B \cdot c_0^2 \cot c_1 \csc c_1 & 0 & 0 & 0 \\ p \cdot c_3 & 0 & 0 & p \cdot c_0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & -\csc^2 c_1 & 0 & 0 & 0 \end{pmatrix} \quad . \tag{1.6}$$

Transformation of covariance matrix

is now calculated in first order of the Taylor expansion (1.4) by

$$Cov_{\text{ZEUS}}(\vec{c}) = J \cdot Cov_{\text{CMS}}(\vec{c}) \cdot J^T \quad . \tag{1.7}$$

1.3 The Implementation

1.3.1 General code-structure

The reconstruction function `findVertices(...)` is designed to get as input-parameter an event and to provide refitted jets. She consists, as shown in the flowchart in figure 1.1, out of three stages: finding the highest momentum-tracks as possible jets-roots, associating tracks to the jet-roots and running of the revertexing-algorithm. For the jet-

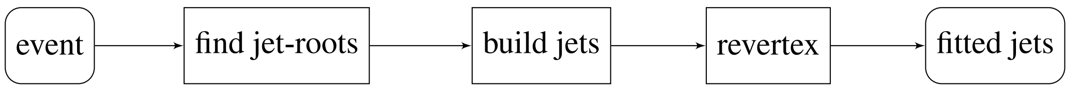

Figure 1.1: general flowchart of `findVertices(...)`

reconstruction a `jet_template` (fig. 1.2) is hold available to have easy and clear access to the jets. The ability to store all related jet-tracks is implemented via a `Jet_TrackMap`.

```
    struct jet_template{
        Int_t Jet_id;
        Int_t Trk_id;           // 0
        Int_t Jet_nr;           // 1
        Float_t Jet_phi;        // 2
        Float_t Jet_eta;
        Float_t Jet_cot_theta;  // 3
        Float_t Jet_et;         // 4
        std::map <Int_t,Double_t> Jet_TrackMap;// ID, p
    };
```

Figure 1.2: Template for jet-storage.

1.3.2 Find jet-roots

The used expression jet-root stands for a high momentum track and is the first reconstruction-step. The idea is to search for the highest momentum tracks that are not too close to each other, means they have a minimum angle of $dR > 1\,\mathrm{rad}$ between each other.

The jet-finding-loop iterates over the user defined maximum of possible jets (`maxJetAnz`)[1]. For every possible jet, see flowchart in fig. 1.3, a second loop iterates over all tracks in the event to find the highest possible momentum track. Every time there is a track with higher momentum a jet-validity-check is performed. To be valid, the track has to has a minimum angle of $1\,\mathrm{rad}$ to all the already stored jets. If so, a dummy-jet of the type `jet_template` gets updated with the track-data and finally stored in a list called `listOfJets`.

For validation the code uses tree flags. The flag `setJet` is set to true if the track passes at least one angle-check with the already stored jets. To prevent that a false track gets stored, the flag `oneDismatch` is set to true to skip updating the dummy-jet, if the track doesn't pass at least one angle-check. The third flag `storeJet` is set to true, if the `jet_template` gets updated.

[1]`maxJetAnz` - can be set in preamble

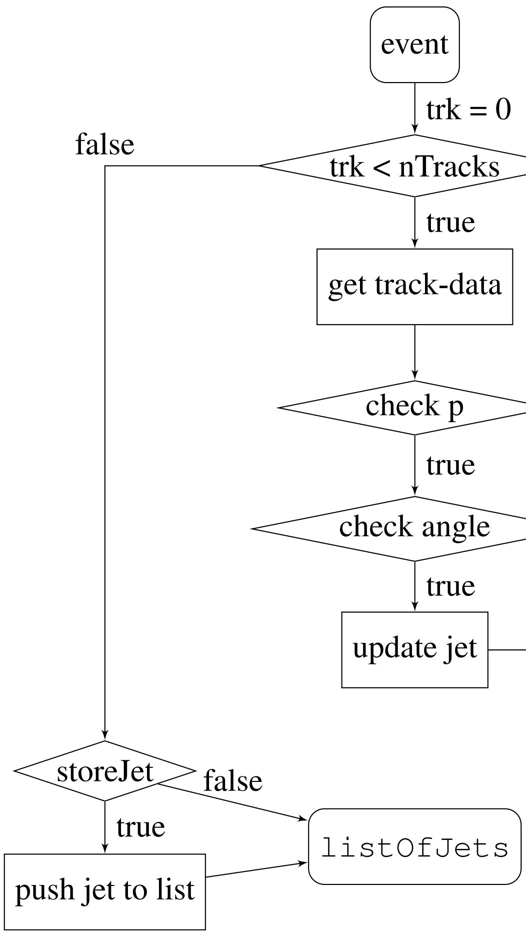

Figure 1.3: flowchart of 'find jet-roots'. Shown is the loop for one jet.

1.3.3 Build jets

After the algorithm has finished to search for jet-roots it starts associating tracks to the jet and stores them to the `Jet_TrackMap` of the specific jet in the `listOfJets`.

As shown in the flowchart of figure 1.4 every track gets checked for a minimum of 4 hits in the Micro-Vertex-Detector (MVD) and a minimum p_T of $0.5\,\mathrm{GeV}/c$. After that a loop over all jet-roots searches for the jet with the smallest distance, but with a maximum angle of $1\,\mathrm{rad}$ to the others. Every time it finds a better one, `jet_min`, an iterator to the actual nearest jet gets updated and the flag `goodJet` gets set to true. The flag ensures, that the track is stored to the nearest jet, if there has been one.

Unfortunately at this point the `jet_template` isn't fully integrated. Currently the algorithm uses next to the template an additional map to store the tracks to the specific jets.

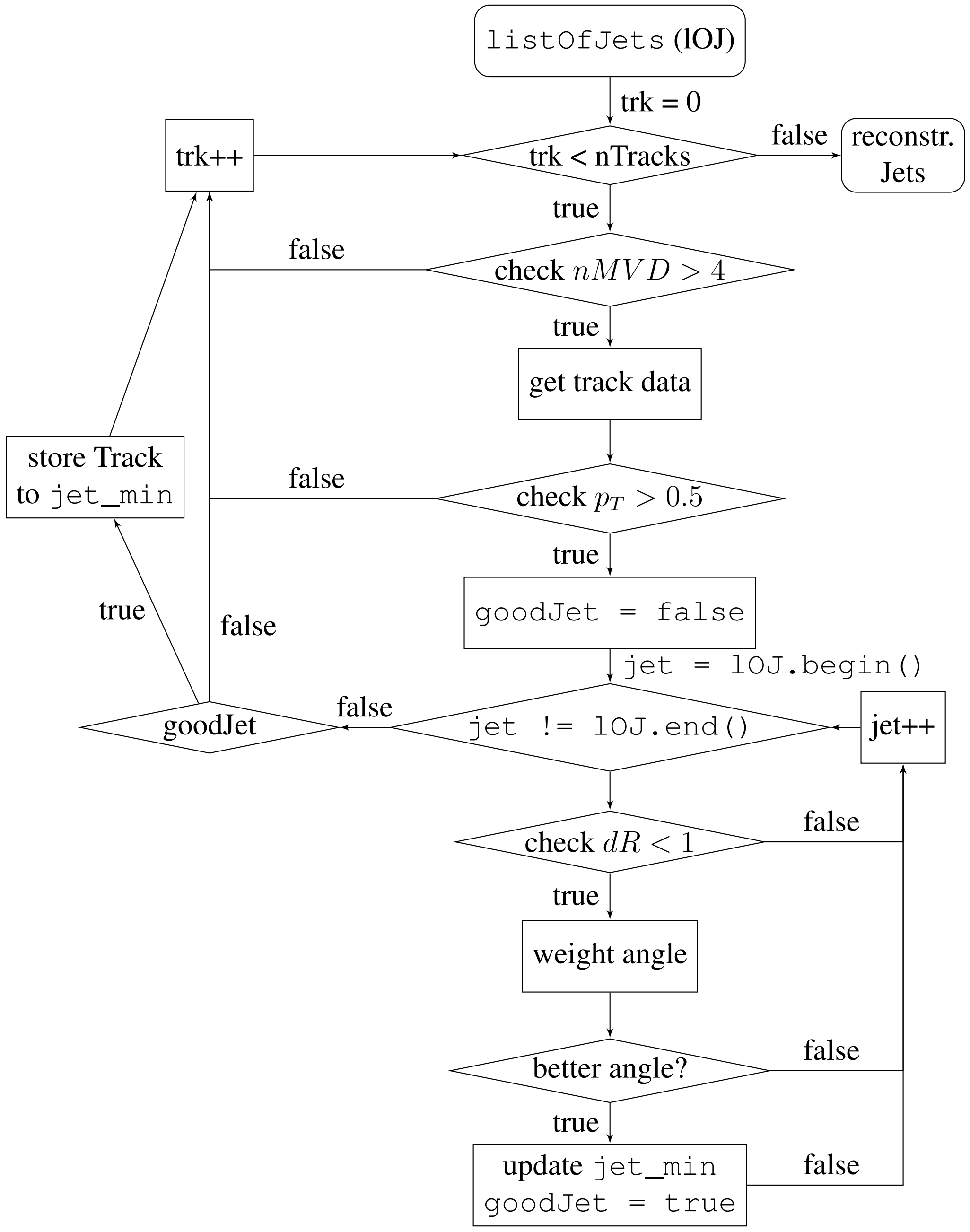

Figure 1.4: flowchart of 'build-jets'; `jet_min` is an iterator of the curr. closest jet.

1.3.4 Revertex

The revertex-step uses the `listOfJets` that got filled with associated tracks from the build-jet-part.

To run a revertex on all jets of the event (see flowchart of fig. 1.5), the algorithm iterates over every jet in the `listOfJets`, gets the specific `jetID` and list of tracks, witch has to has more than 1 track to be processed else it jumps to the next jet in the list. As mentioned before the algorithm uses an additionally a map to store the tracks of a jet. Currently the `jetID` and track-list are read out from this map.

After getting the track-list the algorithm iterates over every track and stores its helix-parameter, covariance matrix and all the other necessary jet parameters to a, for the class-object `TVertex` readable, array. At this point the flag `isCMS` is used to decide if the

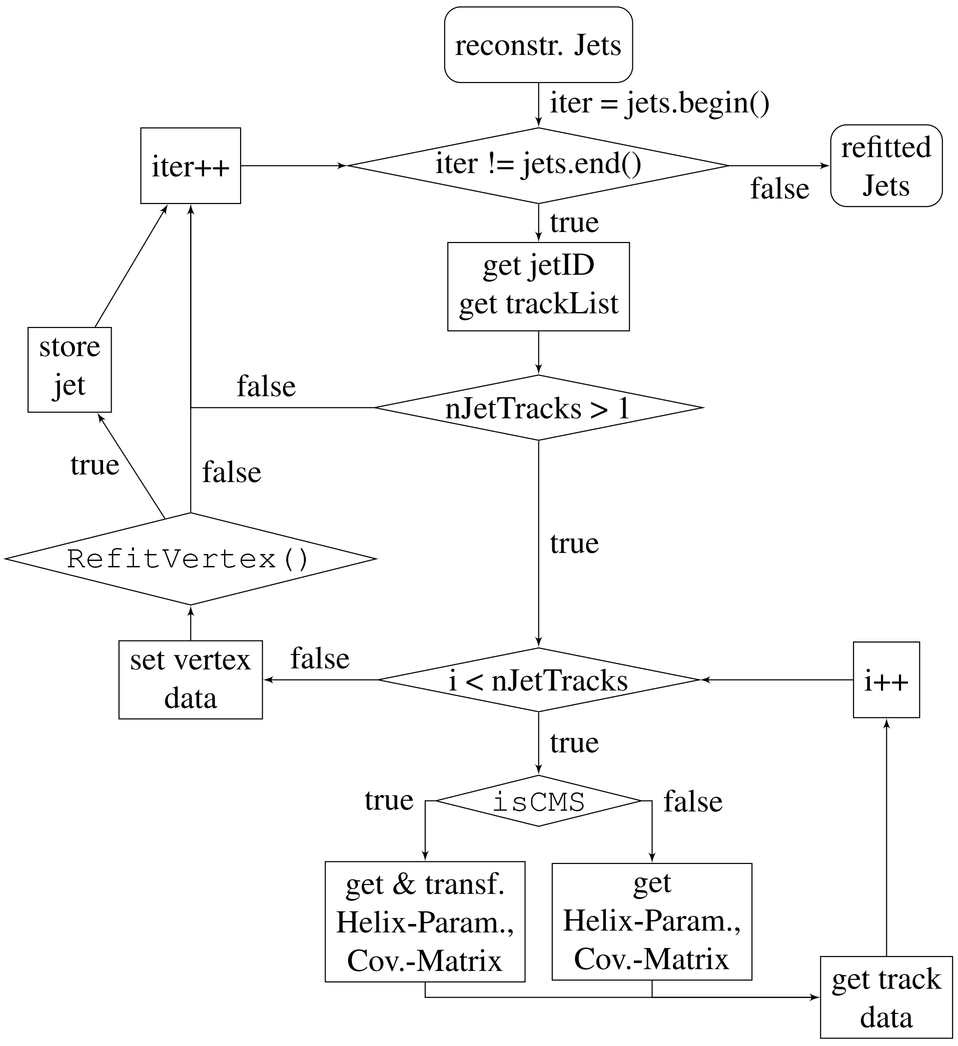

Figure 1.5: flowchart of 'revertex'.

user runs on ZEUS- or CMS-data. If the the flag is set to `true` the code transforms the helix-parameter and covariance matrix to the specific ZEUS-parametrisation.

Then these parameters get set to this `TVertex`-Variable and a refit is performed. `RefitVertes()` tells the user with a bool if the fit was successful or not, thus this return is used to decide to store the refitted values or not. After this the loop does the same for the next jet in the list.

1.4 Flags, Variables and additions (Summary)

The following table 1.3 is a summary of useful flags, variables and additions that can be set in the preamble.

	what it does
`debug`	Enables printout of build-jet to show MVD-hits, p_t, dR, $d\eta$, $d\Phi$ and acceptance or not
`printJets`	Prints out all found jets and associated tracks for the specific event, when refitting was successful
«	Overload of `std::cout`-«-operator to output a `jet_template`; used if `printJets` is enabled.
`isCMS`	Set `true` to switch from ZEUS to CMS
`maxJetNr`	Sets maximum of possible jets. The default-value is 10.

Table 1.3: Summary of useful flags, variables and additions that can be found in the preamble.

1.5 Example-run for ZEUS-event

To take a look into how the code works we take a look into some output of one event. The used event, shown in figure 1.6, has the runnumber 61792, the eventnumber 12967 and consists out of four tracks.

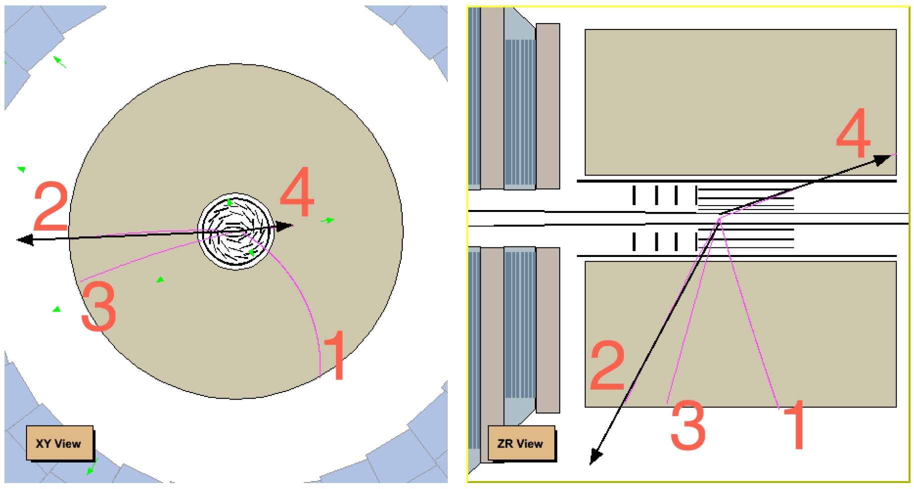

Figure 1.6: Eventdisplay for runnumber 61792 and eventnumber 12967. The red numbers show the specific ID's of the tracks shown in purple. The red arrows show the from the `orange`-framework found jets.

`debug`-output

Running the program with the `debug`-flag enabled we can see, that the build-jet-part works as it should. The output, shown in figure 1.7, shows the check of the track with ID $= 3$ with the 3 jet-roots. As one would expect the track gets associated to the jet with jet-root-ID $= 2$.

`printJet`-output

The output produced with the `printJet`-flag enabled is shown in figure 1.8 and shows the successful refitted jets of the event. In comparison with the eventdisplay of figure 1.6 we see that the algorithm correctly associated the tracks with ID $= 2, 3$ to one jet and left the other two tracks separated.

comparison to `orange`

Thus the last is to check how well the refit of the jet was. A comparison of the secondary vertices and χ^2 of the fit produced by `orange` and this code can be seen in the output shown in figure 1.9. Thus there is no discrepancy in the numbers and the found jets we can say that the algorithm works as it should.

```
TRK–Check :: trk_id = 3 | nMVD = 4
                        | pt = 1.92742
          –>root_id = 4 | dR = 4.01577
                        | dEta = 2.64921 | dPhi = 3.01796
                           –> reject – too far
          –>root_id = 2 | dR = 0.391355
                        | dEta = –0.295773 | dPhi = 0.256276
                        | dR gew = 0.198131 | dRmin = 99999
                           –> store
          –>root_id = 1 | dR = 2.57989
                        | dEta = 0.875102 | dPhi = 2.42694
                           –> reject – too far
```

Figure 1.7: `debug`-output for associating one track to the found jet-roots.

```
tata  :: found Jets:
         Jet_id = 4 | Jet_nr = 3
         Jet_id = 2 | Jet_nr = 1
                    | Trk_id = 2 | Trk_pt = 1.97523
                    | Trk_id = 3 | Trk_pt = 1.92742
         Jet_id = 1 | Jet_nr = 0
```

Figure 1.8: `printJet`-output after the successful refitting of the jets. The algorithm found 3 jet-roots and one jet consisting out of 2 tracks.

```
Orange :: jet = 0 | RunNr = 61792 | EvtNr = 12967
        : SecVert: xor = 1.74319 | yor = 0.239931 | zor = 16.7953
        : Chi2 = 1.888
Debug  :: jet = 1 | RunNr = 61792 | EvtNr = 12967
        : SecVert: x = 1.74319 | y = 0.239931 | z = 16.7953
        : Chi2 = 1.888
```

Figure 1.9: Output of the coordinates of the secondary vertices and χ^2 of the fit produced by `orange` and the reconstruction-code.

1.6 Outlook and Summary

The report showed the progress in the reconstruction of secondary vertices from tracking data. An easy to use `jet_template` has been implemented, the code has been restructured and the transformation from ZEUS- to CMS-parametrisation has been checked and fixed.

Further it was possible to reconstruct jets and vertices from ZEUS-data as it has been done by `orange`. A next step might be to perform some more cross-check also on events with a higher number of tracks to ensure the stability of the code. Then there is the possibility to make use of the already implemented option to run on CMS-data and perform some checks for specific events as it has been done for ZEUS-data.

Last but not least, one could think of a replacement of the old track-map being used in the build-jet section by the new implemented `jet_template` to improve the workflow and reduce memory-usage.

2. $t \to$ 3-jet reconstruction from ZEUS-data at $\sqrt{s} = 318\,\mathrm{GeV}$

2.1 Introduction

The following part of the report shows the progress of a very basic analysis concerning the production of t-quarks during the ZEUS-experiment. As the impact-energy of the electron and proton has been $\sqrt{s}_{\mathrm{ZEUS}} = 318\,\mathrm{GeV}$ and the mass of a t is $m_t = 173.34 \pm 0.27 \pm 0.71\,\mathrm{GeV}/c^2$ there is enough energy that one t could be produced.

2.2 Production and Decay

This analysis looks for the decay $t \to W\ b \to$3-jet. In figure 2.1 a possible feynmangraph[1] of the production of a t is shown. The electron decays over weak interaction to an neutrino and the proton over strong interaction to an not defined particle. This produces a $\bar{t}$ and a b or the other way round a $\bar{b}$ and t, where the antiparticle goes over to the particle.

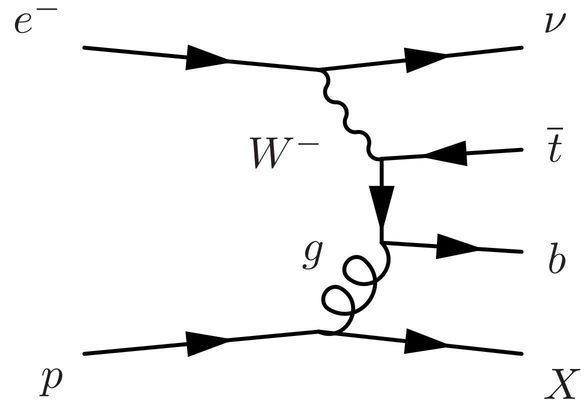

Figure 2.1: Possible feynmangraph of a t-production in an ep-collision at ZEUS.

[1]The probability says, that there will be half of a t in the whole data being produced with the ZEUS-ecperiment.

Because of the t decaying in 3 jets and the production of a b there should be at least 4 jets in the detector and because of the production of a neutrino a cut on missing energy would improve the results, but because of it being a basic analysis only the reconstruction of 3 jets to one t has been minded.

A possible t decaying into 3-jets can be seen in figure 2.2.

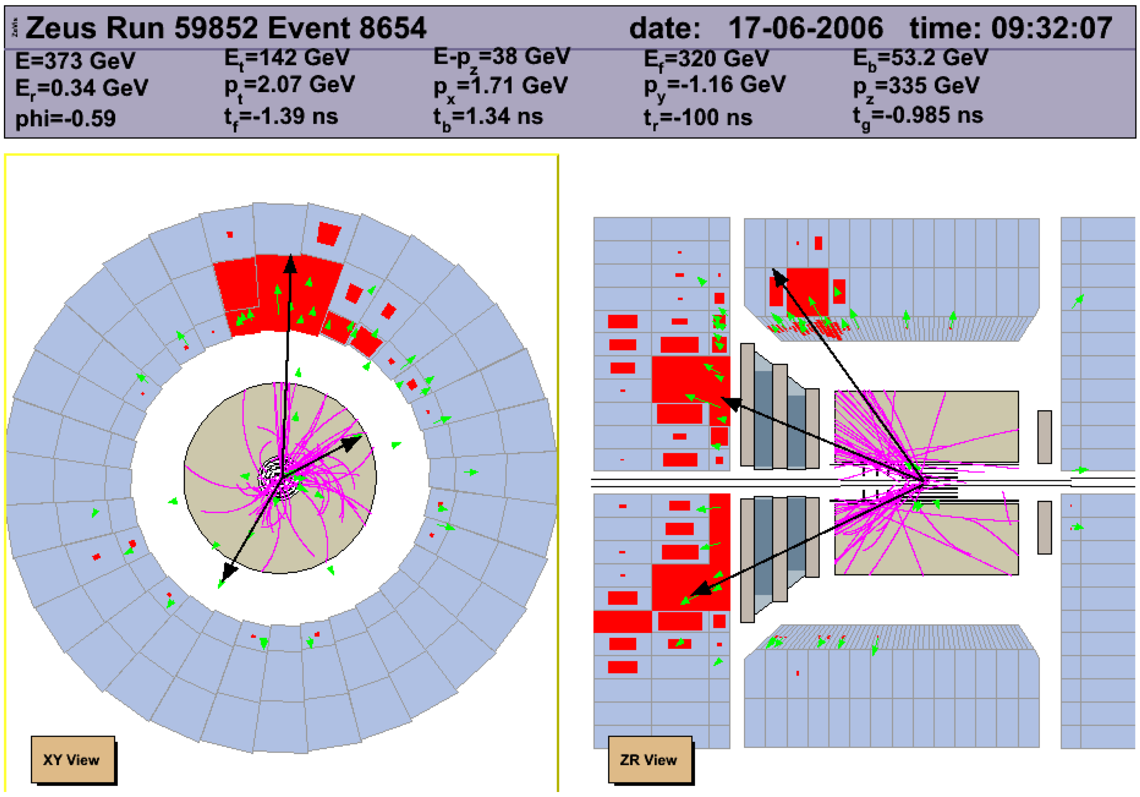

Figure 2.2: Possible decay of $t \rightarrow$3-jet at ZEUS.

2.3 Data and Cuts

The analysis used version-`v08b` files. For data the years `03p`, `04p`, `06e`, `06p`, `07p`, `ler` and `mer` have been used. The used MC[2]-files have been 'Low Q2 Ariadne QCD NC sample'-files of the year `0304p`. Jets-types that have been used were A and B. Both types are massive Kt-jets, but type-A is a jet out of a photoproduction (PHP) and type-B out of deep-inelastic-scattering (DIS). The used cuts for the jets have been $E_t > 5\,\text{GeV}$ and $|\eta| < 2.5$ As the electron decays to a neutrino most t should be seen in the graphs for PHP.

2.4 Results

The figures 2.3,2.4,2.5 and 2.6 show the kinematic results and figure 2.7 show the invariant mass of the reconstruction. As the MC has been for DIS the red curve on the PHP graphs has to be ignored. Interesting is the peak that can be seen for the MC for the invariant mass (fig. 2.7). However the invariant mass for PHP-data shows no peak in the interesting area around $171\,\text{GeV}/c^2$.

[2]MonteCarlo

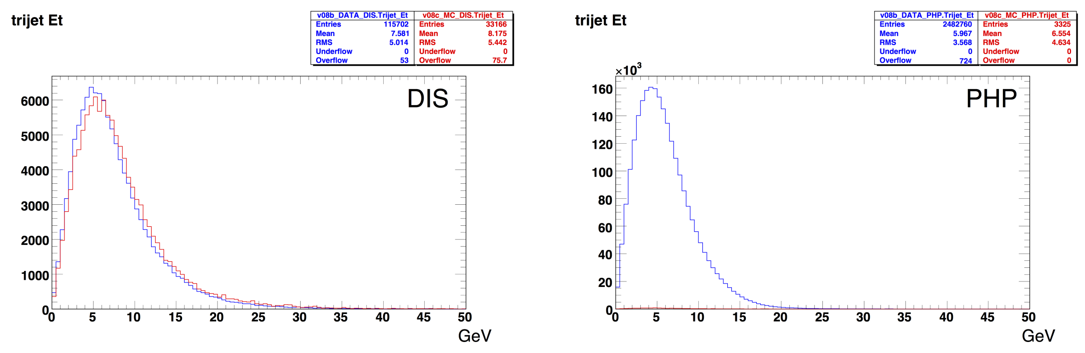

Figure 2.3: E_t of the 3-jet-particle. As the MC has been for DIS the red curve on the PHP graph has to be ignored.

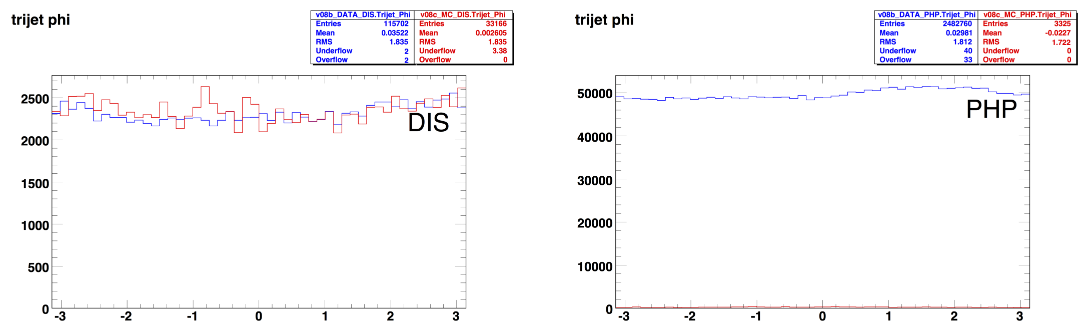

Figure 2.4: Φ of the 3-jet-particle. As the MC has been for DIS the red curve on the PHP graph has to be ignored.

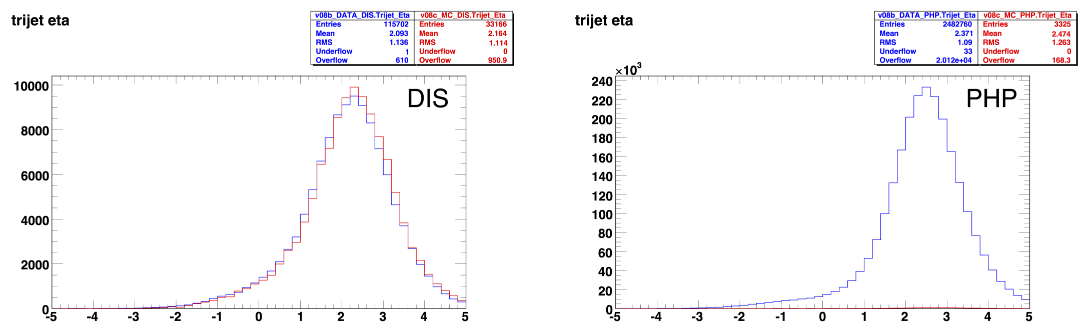

Figure 2.5: η of the 3-jet-particle. As the MC has been for DIS the red curve on the PHP graph has to be ignored.

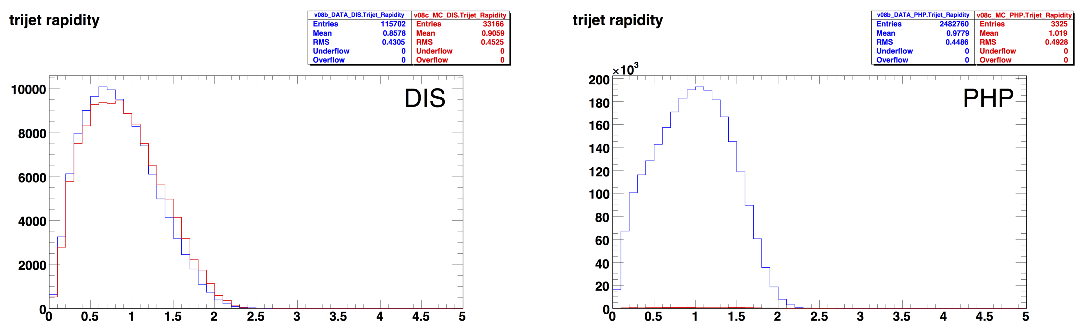

Figure 2.6: Rapidity of the 3-jet-particle. As the MC has been for DIS the red curve on the PHP graph has to be ignored.

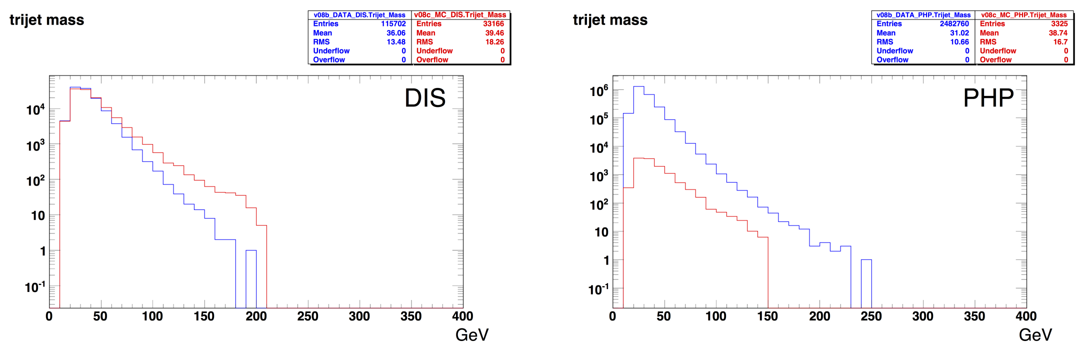

Figure 2.7: $m_{t,\text{inv}}$ of the 3-jet-particle. As the MC has been for DIS the red curve on the PHP graph has to be ignored.

2.5 (The Unanalysed)

To get better results a cut on the E_t of the event has been performed, because a look into the eventdisplay for events out of the t-mass region showed that there has been events with jets of higher E_{t_jet} than the total E_{t_evt} of the event. Thus every event with $E_{t_\text{jet}} < E_{t_\text{evt}}$ got skipped. Cause of the fact that except of the years `ler`, `mer` and `03p` every job broke with a `bus`-error in the end, no root-file has been produced for them.

However the output of the interesting events, that got stored in text-files, was successful. The summarised output can be seen in table 2.1. An interesting event out of the year `06p` with the runnumber 61296 and eventnumber 12452 can be seen in figure 2.8.

year	case	type	mass	n_{jets}	run-nr.	event-nr.
07p						
	2	PHP	213.448	4	61974	15818
	1	PHP	170.375	3	62383	33060
06p						
	1	PHP	174.355	3	60689	4559
	1	DIS	170.727	3	60871	10467
	1	PHP	179.531	3	60887	34356
	1	PHP	171.114	3	60939	202734
	1	PHP	171.882	4	61296	12452
06e						
	1	PHP	171.895	3	59027	53122

Table 2.1: This table shows the output of possible invariant masses for a t. Case 1 stands for the region $170\,\text{GeV}/c^2 < m < 180\,\text{GeV}/c^2$ and case 2 for $m > 200\,\text{GeV}/c^2$

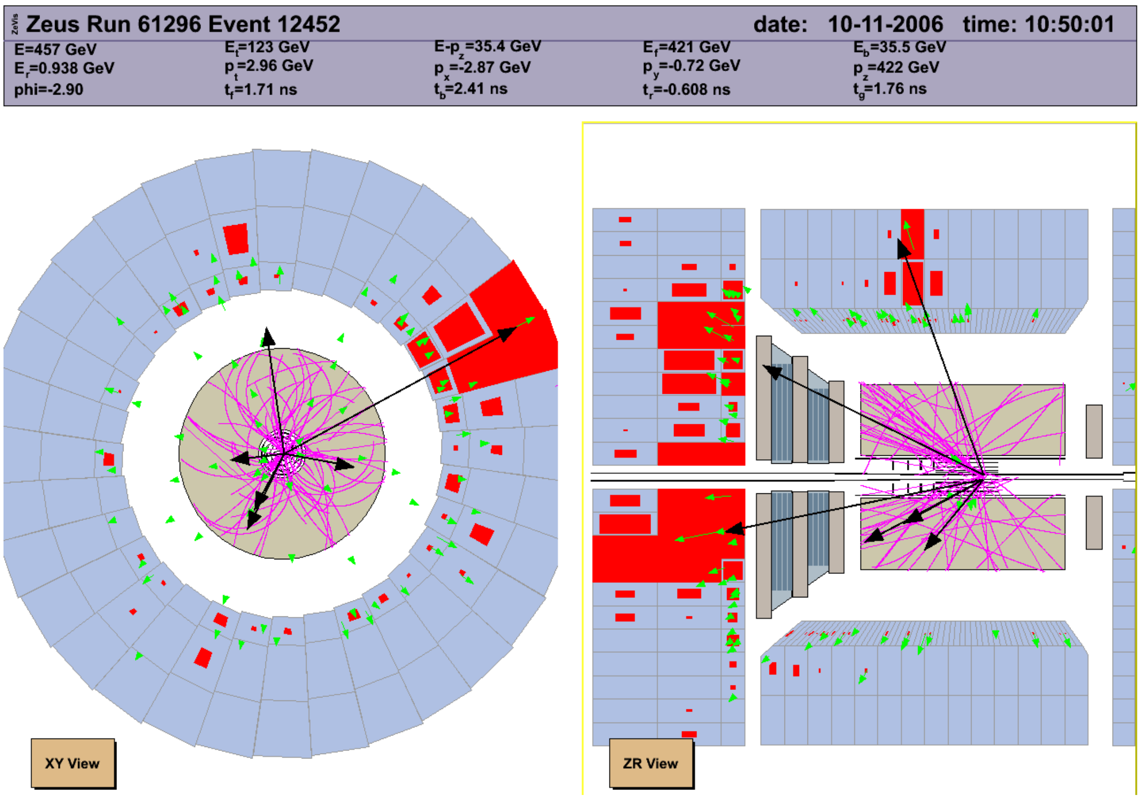

Figure 2.8: Interesting but unanalysed event with $m_{inv,3-jet} = 171.882$.

2.6 Summary and Outlook

The results (vgl.fig. 2.7) didn't show us a peak in the expected mass-region of the t. This may be caused by minding a cut on the missing energy of a produced neutrino and looking for events that have at least 4 jets. A look into data with these cuts may show a produced t, as the briefly look for a cut on E_t for jet- and total event-energy in section 2.5 (The Unanalysed) showed.

Bibliography

[1] L. van der Schaaf, “Secondary vertex reconstruction from cms using zeus title libray,” tech. rep., 2013.

[2] V. Roberfroid, “Improvement of the Kalman Filter Fit of the ZEUS experiment,” `arXiv:physics/0701271 [PHYSICS]`.

[3] A. Strandlie and W. Wittek, “Propagation of covariance matrices of track parameters in homogeneous magnetic fields in CMS,”.