Pattern Recognition
in the
ZEUS Central Tracking Detector

David Shaw
University College London

Submitted to the University of London
for the degree of
Doctor of Philosophy
October 1990

Abstract

The HERA accelerator will collide 30 GeV electrons with 820 GeV protons. The ZEUS and H1 experiments are currently being prepared to study the resulting interactions.

At the heart of ZEUS, the Central Tracking Detector (CTD) is a large cylindrical drift chamber. Using a mixture of axial and stereo sense wires it will allow the reconstruction of charged particle tracks in three dimensions.

The first step in full CTD track reconstruction will be to find tracks in just two dimensions using information from the axial sense wires only. In normal operation this two dimensional pattern recognition problem must be solved separately at two different stages of processing: a fast online system must provide information for trigger processing, while later a highly efficient offline system must allow as much information as possible to be extracted from the recorded events.

The CTD online pattern recognition system is part of the ZEUS second level trigger (SLT) and is required to process one event every millisecond. Our solution to this difficult problem involves a highly parallel algorithm running on a large array of transputers. Our discussion of this question is broadened to include similar applications in the readout and trigger systems of future detectors.

The CTD offline pattern recognition methodology bases track finding on the extension towards the interaction point of seed tracks found in the outer layers of the CTD. A high degree of software engineering ensures that our design and implementation is both robust and flexible. Several complementary options for finding seed tracks are supported, including one specially designed to exploit vector architectures.

As one possible use of the CTD in early physics analysis a study is made of using tracking information to help separate the neutral current and charged current event classes.

Peter and Rosina

Acknowledgments

I would like to thank the many people without whom this thesis would not have been possible: R.Belusevic, R.Devenish, T.W.Jones, J.B.Lane, D.J.Miller, G.Nixon, D.H.Saxon, and all the software team at UCL, as well as all my other colleagues within UCL and ZEUS.

My special thanks must go to my supervisor F.W.Bullock and to K.Long and D.Gingrich who have struggled hard to continue my education, trying to teach me the skills needed by an experimental physicist. Hopefully their efforts, and those of my tutors at St. Chad's school Runcorn and Balliol College Oxford, have had some success.

Finally I would like to thank my family and friends, especially Christine Kelly and my mother Rosina Shaw who have helped me in so many ways for so long.

Foreword

Pattern recognition is a broad subject, and for a long time physicists have made use of such techniques to help in the analysis of their experimental data. The rapid advance in modern computing technology means that this base of knowledge must be continually updated and adapted, incorporating new ideas and techniques. In this thesis I shall attempt to meet this challenge.

Part I is a brief introduction to the physics of HERA and the design of the ZEUS detector. Only after the reader is familiar with some of this general material can we settle down to discuss the real point of the thesis: the use of pattern recognition techniques to find tracks in the CTD. This complex task must be performed in both the online and offline systems.

Part II discusses track finding in the online environment, tracing the development of the CTD SLT through each of its early design phases. It is instructive to compare this pattern recognition task to other real-time image processing applications: typically these might take a high resolution image (512×512 pixels) and search for objects (straight edges). To deceive the human eye, the processed information would have to be refreshed at a rate of about about 50Hz. The CTD SLT system must take the raw data from each CTD event ($4608 \times 50 \times 2$ pixels) and find 2D tracks (circular trajectories). This has to be accomplished at a rate of 1kHz.

Part III investigates the different problems posed by the offline environment. Here our pattern recognition techniques must achieve a high level efficiency, extracting as much information as possible from each event. The history of the CTD track reconstruction system is charted through the many stages of its design and re-design. A particular emphasis is placed on the development of the 2D track finder, and only in the final chapters of this thesis do we demonstrate that this important part of the system has an acceptable performance.

The careful reader, trying to contrast the online and offline environments, will notice that parts II and III of this thesis have been structured to emphasise their common themes. He will also notice that the same analysis techniques and development tools have played a role in both cases. For many, the online/offline comparison will be the most interesting aspect of this thesis.

It is regrettable that I am only able to present one short chapter reporting a physics study undertaken on my own initiative. This is a sad reflection of the fact that the completion of HERA has been delayed by over a year during the time that I have been a graduate student.

David Shaw
October 1990

Certification of Conjoint Work

This thesis represents conjoint work carried out by my postgraduate student David Shaw as part of the ZEUS high energy physics collaboration.

David has played a leading role in the development of pattern recognition techniques for the ZEUS Central Tracking Detector (CTD). Uniquely, he has been involved in both the online and offline working groups.

Within the CTD online group David has made several original contributions. These include several feasibility studies based on microprocessor performance, a comparative study of possible online segment finding algorithms, and an important part in the evaluation of the final preferred solution. He has continued to make original contributions in this field by extending his work towards applications at future accelerators.

Within the CTD offline group David has played an equally important role, with many original contributions centering on the development of a 2D track finding algorithm suitable for use at ZEUS. This algorithm, based on a conformal mapping technique, has the special advantage that it is highly suitable for implementation on vector machines. His part in this project has led David into playing a driving role in the design and evaluation of the final CTD offline reconstruction program.

Working in both the online and offline groups has allowed David to transmit ideas freely between the two. This has been invaluable to both groups, easing the sharing of common methods and tools. An example of this was seen in the development of a technique for the evaluation of pattern recognition performance. David played a large part in this project, which was later used by both the online and offline groups.

David has rounded off his work by pursuing another original idea of his own. He has investigated how tracking information may be used to help the identification and separation of the neutral current and charged current event classes in early ZEUS physics analysis.

Prof. F.W.Bullock
Supervisor
October 1990

Contents

Part I

Introduction

Chapter 1

Physics at HERA

HERA will collide 30 GeV electrons with 820 GeV protons. Integrated luminosities of over 100pb^{-1} per year are expected. Plans exist to provide electron beams of either helicity, and to accelerate positrons as well as electrons. The main accelerator parameters are summarised in table 1.1.

In the past, lepton-nucleon scattering has played a major role in advancing our understanding of the structure of matter. The size of the proton was first measured by electron scattering at Stanford in 1954 [1], while evidence of its composite structure emerged at SLAC in 1968 [2]. Through further experiments at SLAC, CERN and FERMILAB [3] this structure was interpreted in terms of the quark-parton model [4]. This line of research will be continued at HERA by two major experiments, ZEUS [5] and H1 [6].

1.1 Kinematics

Consider an electron (p_e) which collides with a proton (P). Using the four-momentum of these particles we define s, the square of the centre of mass energy,

$$s \equiv (p_e + P)^2$$

Centre of mass energy	314 GeV
Time between bunch crossing	96ns
Luminosity	$1.5 \times 10^{31}\text{cm}^{-2}/\text{s}$
Nominal interaction region	$\sigma_x \sim 400\mu\text{m}$
	$\sigma_y \sim 100\mu\text{m}$
	-20cm $< z <$ 20cm

Table 1.1: Summary of HERA parameters

This is a Lorentz invariant and may be calculated in any convenient reference frame. Let us assume that in our chosen frame the energies of the electron and proton are E_e and E_p respectively. At HERA energies it is normally a good approximation to neglect the electron and proton rest masses, and so

$$s \simeq 4E_eE_p$$

After the collision the electron is scattered as a lepton (p_l). We define $Q^2 = -q^2$, the square of the four-momentum transferred by the collision,

$$Q^2 \equiv -(p_e - p_l)^2$$

Again this variable is a Lorentz invariant. In our chosen reference frame the scattered lepton has energy E_l, and is deflected by an angle θ_l, and so

$$Q^2 \simeq 4E_eE_l \sin^2 \frac{\theta_l}{2}$$

Two other variables are also commonly used,

$$x \equiv \frac{Q^2}{2P \cdot q}$$
$$y \equiv \frac{P \cdot q}{P \cdot p_e}$$

The variables x and y are dimensionless and have a range between 0 and 1. They are often expressed in terms of ν, the energy transfer of the interaction in the proton rest frame. If m_p is the rest mass of the proton, then

$$m_p\nu \equiv P \cdot q \simeq 2E_p \left(E_e - E_l \cos^2 \frac{\theta_l}{2} \right) \tag{1.1}$$

and so we use

$$x = \frac{Q^2}{2m_p\nu}$$
$$y = \frac{\nu}{\nu_{max}}$$

Both these variables have an intuitive interpretation: y measures how 'hard' a collision is as a fraction of that kinematically allowed, while in the quark-parton model, x is the fraction of the proton momentum carried by the struck quark.

If we again neglect the proton rest mass, the variables Q^2, x, and y are related by the equation

$$Q^2 = sxy \tag{1.2}$$

For a given s, we have two free parameters with which to specify the kinematics of an event.

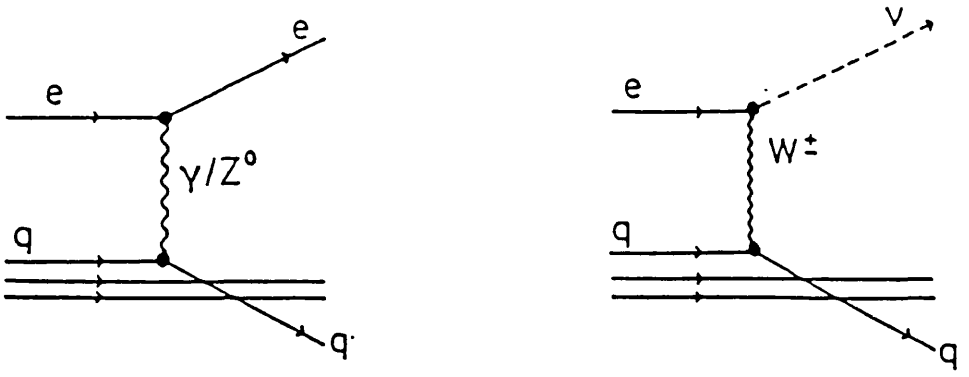

Figure 1.1: Quark-parton interpretation of NC and CC events: an incoming electron (e) interacts with a quark (q) from within the proton

1.2 Deep Inelastic Physics

Deep inelastic lepton-proton scattering has been studied by the EMC and BCDMS collaborations up to $Q^2 \sim 100$ GeV^2 [8, 9]. At HERA the accessible kinematic region will be greatly extended, with statistically useful event samples available up to $Q^2 \sim 4 \times 10^4$ GeV^2 [7].

Within the quark-parton model, deep inelastic electron-proton scattering is interpreted as a collision between an incident electron and a quark from within the proton sub-structure, see figure 1.1. In a neutral current (NC) event, the interaction is mediated by the exchange of a γ/Z^0 boson, while in a charged current (CC) event, the interaction is mediated by a $W^{\pm}$ boson. These two event types are distinguished by the type of scattered lepton present in the final state, an electron in NC events and a neutrino in CC events.

Following [10], we can write down expressions for the NC and CC differential cross sections in the leading order of the standard electroweak theory,

$$\frac{d\sigma^2_{NC}}{dxdQ^2} = \frac{4\pi\alpha^2}{xQ^4}\left[y^2 xF_1 + (1-y)F_2 + \left(y - \frac{y^2}{2}\right)xF_3\right]$$

$$\frac{d\sigma^2_{CC}}{dxdQ^2} = \frac{4\pi\alpha^2}{xQ^4}\left[y^2 xW_1 + (1-y)W_2 + \left(y - \frac{y^2}{2}\right)xW_3\right]$$

where F_i and W_i are each functions of x and Q^2, and represent the structure functions of the proton for NC and CC events respectively. These are different, as in this representation they must reflect the internal details of each coupling.

(pb)	σ_{NC}	σ_{CC}
$Q^2 > 10^2$ GeV^2	5300	65
$Q^2 > 10^3$ GeV^2	220	44
$Q^2 > 10^4$ GeV^2	3.5	4.6

Table 1.2: NC and CC cross sections at HERA.

Table 1.2 uses [11] to integrate these differential cross sections over typical Q^2 ranges accessible at HERA. These calculations have to extrapolate the proton structure functions from present data, in accordance with theoretical models. The direct measurement of these functions is one of the principle physics goals of HERA.

Comparing the results of structure function measurements with prediction offers an opportunity to confirm current theories and to search for physics beyond the standard model. We may

- Probe the proton for further sub-structure down to 10^{-18}m (10^{-20}m if polarisation assymetries are used).
- Make a detailed study of the electroweak interaction, using polarised beams to search for right-handed charged currents and new exchange bosons.
- Test the QCD predictions for the evolution of the structure functions with Q^2.
- Use multi-jet final states to test perturbative QCD calculations and measure Λ_{QCD}.

An accurate knowledge of the high Q^2 structure of the proton will be essential while designing the next generation of accelerators.

1.3 Photoproduction Physics

At very low values of Q^2 the cross section for electron-proton scattering via photon exchange becomes relatively large ($\sim$ 100μb). Such interactions are usually referred to as photoproduction, since the $Q^2 \simeq 0$ GeV^2 photons behave like a quasi-real photon beam.

The effective rate of these events is suppressed by the finite detector acceptance: the incident electron suffers only a small deflection, and at very low Q^2 can only be detected if special apparatus is available further down the beam line. Three different mechanisms can give rise to events with high transverse momentum (P_t) particles in the final state [12]:

1. Point-like couplings between the photon and a quark or gluon from within the proton. These processes are known as QCD Compton scattering and photon-gluon fusion respectively.

2. Point-like couplings between a photon constituent and a quark or gluon from within the proton.

3. Vector meson dominance, where the photon behaves like a vector meson during its collision with the proton. Though this mechanism has the largest total cross section, it produces relatively fewer events containing high P_t particles.

Each of these mechanisms can produce one or more jets of high P_t particles in the final state. These events are an important background to deep inelastic physics in the region $Q^2 < 500$ GeV^2 [12], but also provide interesting physics in their own right.

The production of charm and bottom quarks at HERA is dominated by photoproduction via photon-gluon fusion [13]. The study of heavy quark physics has many areas of interest:

- Using J/ψ production to measure the low x gluon distribution of the proton.

- A direct observation of time dependant oscillations in the B^0-$\bar{B}^0$ system.

- Searching for rare decays of charmed and bottom hadrons.

Attempting to measure the gluonic content of the photon structure function is just one other area of interest in photoproduction physics.

1.4 Exotic Physics

The new energy domain accessible at HERA provides an opportunity for the discovery of new physics processes. Some of the possibilities have already been discussed in the two previous sections. The remaining major areas of interest are grouped together in this section.

Leptoquarks have been proposed in many theoretical models which try to unify the strong and electroweak forces [14]. These particles carry the quantum numbers of both leptons and quarks and could be formed as resonances in electron-proton collisions. This would provide a very clean signal for discovery at HERA, provided their mass was not above threshold, $\sqrt{s} = 314$ GeV. Leptogluons might be discovered in a similar way, though these objects are more theoretically speculative.

Excited states of electrons also provide exceptionally clean signals at HERA. Elastic production of these states ($ep \rightarrow e^*p$) would leave the proton little

deflected, while the excited electron would decay leaving only an electron and a photon observable at high P_t [15]. Again discovery is limited by the kinematic limit, $\sqrt{s} = 314$ GeV.

Supersymmetric (SUSY) models characteristically predict the existence of scalar partners for all the known quarks and leptons. The production of these particles at HERA can produce clear signals based on event topology and missing energy. It is normally assumed that any SUSY particles produced decay to quarks, leptons and photinos. The latter escape detection, unbalancing the observed event in the transverse plane [16]. The possibilities for the detection of SUSY particles at HERA depend heavily on the exact model used [17].

After the identification of a new physics process careful experimentation is needed to unravel the structure of the interaction. At HERA this will be greatly assisted by the use of polarised beams, and the ability to accelerate positrons as well as electrons.

1.5 Summary

HERA is an exciting new accelerator facility, offering the scope for a wide range of experimental activities. Once again, electron-proton collisions will play a major role in the advancement of physics.

Bibliography

[1] R.Hofstadter, R.W.McAllister, Phys. Rev. 98 (1954) 217.
R.Hofstadter, 'Nuclear and Nucleon Structure', Frontiers in Physics, W.A.Benjamin Inc., New York 1963.

[2] W.K.H.Panosky, XIV International Conference on HEP, Vienna, Austria 1968, (CERN scientific information service, Geneva, Switzerland 1968) p.23.
M.Breidenbach et al, Phys. Rev. Lett. 23 (1969) 935.

[3] For review see G.West, Phys. Rep. 18C (1975) 263.

[4] R.P.Feynman 'Photon-Hadron Interactions', Frontiers in Physics, W.A.Benjamin Inc.,Readding, Massachusetts,1972.
F.E.Close 'An Introduction to Quarks and Partons', Academic Press, London 1980.

[5] ZEUS collaboration, letter of intent 'ZEUS - a detector for HERA', DESY June 1985.
ZEUS collaboration, 'The ZEUS detector - a technical proposal', DESY March 1986.

[6] H1 collaboration, letter of intent 'A letter of intent for an experiment at HERA', DESY June 985.
H1 collaboration, 'Technical proposal for the H1 detector', DESY March 1986.

[7] G.Ingelman 'Deep Inelastic Physics at HERA' DESY 87-144 (1987).

[8] EMC collaboration: J.J.Aubert et al, Nucl. Phys. B272 (1986) 158: B293 (1987) 704.

[9] BCDMS collaboration: A.C.Benvenuti et al, CERN preprint, submitted to Int. EPS Conf. on HEP, Uppsala (1987).

[10] J.Blumlein et al, 'Structure Functions, Quark Distributions and Λ_{QCD} at HERA' HERA workshop vol.1 pg.67.

[11] G.Ingelman, LEPTO version 5.2, DESY.

[12] S.J. de Jong 'Hard Scattering of (almost) Real Photons at HERA' HERA workshop vol.2 pg.533.

[13] A.Ali et al, 'Heavy Quark Physics at HERA' HERA workshop vol.1 pg.395.

[14] J.Bijnens 'Leptoquarks and Leptogluons at HERA' HERA workshop vol.2 pg.819.

[15] Ch.Berger et al, 'Excited Electrons at HERA' HERA workshop vol.2 pg.813.

[16] R.J.Cashmore et al, Phys. Rept. 122C (1986) 277.

[17] J.Bartels 'SUSY at HERA - Theoretical Expectations' HERA workshop vol.2 pg.863.

Chapter 2

The ZEUS Detector

The ZEUS collaboration is preparing a large general purpose detector for use at HERA [1]. The main features of this detector are summarised in table 2.1.

Calorimetry	$2.2° < \theta < 176.5°$
	$\sigma(E)/E = 0.35/\sqrt{E} \oplus 0.02$ (Hadrons)
	$\sigma(E)/E = 0.17/\sqrt{E} \oplus 0.01$ (Electrons)
Charged particle tracking	$7.5° < \theta < 170.0°$
	$\sigma(p)/p = 0.0021p \oplus 0.0029$ at 90°
	$\sigma(b) \sim 50\mu$m (impact parameter)
Electron identification	10^{-4} hadron rejection
Muon identification	10^{-3} hadron rejection

Table 2.1: Summary of ZEUS detector parameters.

2.1 High Resolution Calorimetry

The energy of any particle or jet of particles is measured using a depleted uranium/scintillator calorimeter. The three sections of this high resolution calorimeter (FCAL, BCAL and RCAL) are illustrated in figure 2.1. The combined system is hermetic, having a polar angle coverage $2.2° < \theta < 176.5°$, and an effective depth varying from 7 to 5 hadronic interaction lengths in the forward and rear directions respectively. Containing 499 tons of uranium, this calorimeter absorbs 95% of the energy of 90% of the jets incident on it. The remaining energy is measured by an iron/proportional tube backing calorimeter (BAC).

Energy is deposited in a calorimeter when an incident particle produces a shower of secondaries. Two mechanisms may contribute to this cascade process:

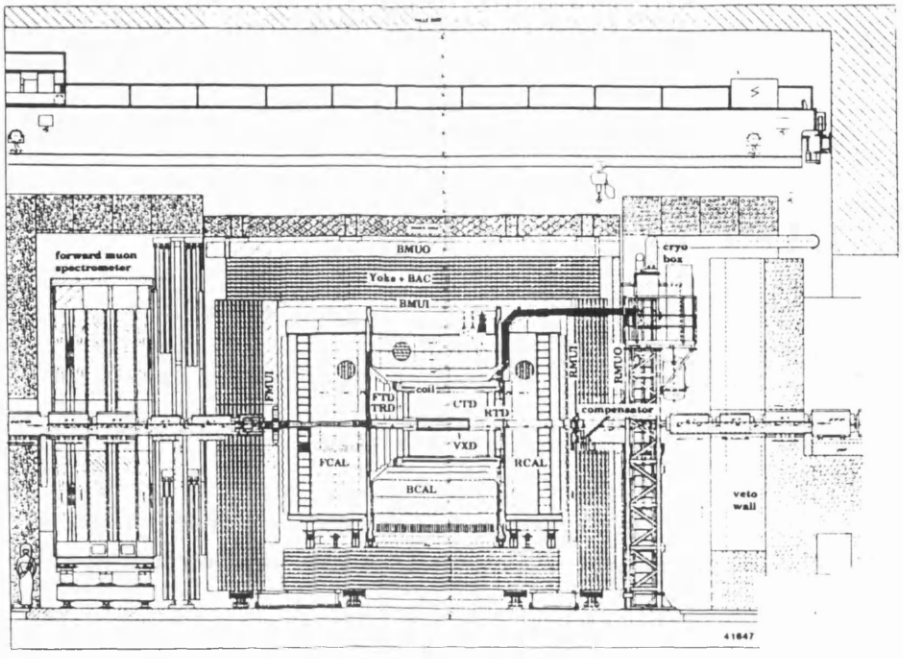

Figure 2.1: The ZEUS detector.

Electromagnetic processes: These tend to have a low multiplicity and occur on the distance scale associated with the electromagnetic radiation length, $X_0 \sim 3.2$mm of depleted uranium.

Hadronic processes: These tend to have high multiplicity and occur on the distance scale associated with the hadronic interaction length, $\lambda \sim 10.5$cm of depleted uranium.

If the incident particle is an electron then electromagnetic processes dominate, the shower develops rapidly and is statistically well behaved. However, if the incident particle is a hadron both these processes may play a part. Unfortunately, calorimeters do not in general offer the same response to each of these mechanisms, and event to event fluctuations in their relative contributions lead to variations in the total energy observed. This typically limits the effective energy resolution for incident hadrons to $\sigma(E)/E \sim 0.5/\sqrt{E}$.

It has been shown [2] that the energy resolution for hadrons can be optimised by adjusting the calorimeter design until its response is approximately equal for electromagnetic and hadronic processes. The calorimeter is then said to be 'compensated'. This has been achieved for the ZEUS calorimeter by carefully selecting the ratio of the thicknesses of the depleted uranium and scintillator plates. For incident hadrons, the energy resolution will be $\sigma(E)/E = 0.35/\sqrt{E} \oplus 0.02$

2.2 Charged Particle Tracking

At the centre of ZEUS, the inner tracking detectors provide charged particle tracking with a polar angle coverage $7.5° < \theta < 170.0°$. This system includes the vertex detector (VXD), the central tracking detector (CTD), the forward detector (FTD), and the rear tracking detector (RTD). These are illustrated in figure 2.1.

The VXD combines the mechanical simplicity of a jet chamber with the high spatial resolution of a time-expansion chamber. A resolution of 35μm is achieved by using a relatively low drift velocity (6μm/ns) over an extended region of constant drift field, followed by a well defined region of amplification near the sense wire. The VXD is cylindrical with inner and outer radii of 100mm and 156mm respectively. It offers an active region 1.5m long containing 12 layers of sense wires running parallel to the beam direction.

The CTD is a large cylindrical drift chamber, presenting 72 layers of sense wires to tracks traversing the whole of its active radius. Spatial resolutions of up to $\sim 100\mu$m are possible. This sub-detector is described in detail in chapter 3.

The FTD and RTD provide extra tracking coverage in the forward and backward directions respectively. These sub-detectors both contain similar drift chamber modules, measuring hits with a spatial resolution of up to $\sim 100\mu$m. Each module is cylindrical, having an outer radius of between 62cm and 123cm, and a thickness of 15cm. The FTD contains three modules, each separated by 21cm along the beam direction, while the RTD contains only one. A track traversing any of these modules, travelling in a forward or backward direction would encounter a total of 18 layers of sense wires, each lying in a plane perpendicular to the beam direction.

The combined inner tracking system has a momentum resolution of $\sigma(p)/p = 0.0021p \oplus 0.0029$ for tracks perpendicular to the proton beam direction and is able to measure track impact parameters to $\sigma(b) \sim 50\mu$m.

A separate outer tracking system is used for muon detection (see section 2.4).

2.3 Electron Identification

When an electron enters the high resolution calorimeter it will deposit most of its energy in the first $\sim$ 25cm. Electron identification makes use of this property, also insisting that the energy cluster in the calorimeter can be matched to a track in the inner tracking system.

To improve the performance of this system for electrons within jets, layers of silicon pads are included within the high resolution calorimeter. In the forward direction, electron identification is further strengthened by transition radiation detector (TRD) modules, located in the spaces between the three FTD modules. In the barrel region, dE/dx measurements from the inner tracking system aid

electron identification at low energies.

The hadron rejection of this combined system is better than 10^{-4} for tracks travelling in the forward direction.

2.4 Muon Identification

Muons are identified as minimum-ionising particles penetrating through the zeus calorimeters. Such tracks are reconstructed by a comprehensive muon tracking system. This consists of three stages:

Inner tracking detectors: Muons are tracked before they enter the high resolution calorimeter by the sub-detectors described in section 2.2.

Inner muon detector: Between the high resolution calorimeter and the backing calorimeter muons are detected using two layers of limited streamer tubes (FMUI, BMUI and RMUI).

Outer muon detector: After the backing calorimeter muons are again tracked using limited streamer tubes (FMUO, BMUO and RMUO). In the forward direction this system is augmented into a complete muon spectrometer.

This is illustrated in figure 2.1.

Initial muon momentum is measured by the inner tracking system, while an external measure of momentum is made between the inner and outer muon chambers. To accomplish this the backing calorimeter is magnetised toroidally, with additional toroids in the forward region. Momentum resolutions of $\sigma(p)/p \sim$ 20% are expected for tracks in the forward direction with $p = 100$ GeV/c. Pion rejections of 10^{-3} should be achieved in this direction up to $p = 40$ GeV/c.

2.5 Detection of Forward Going Particles

The ZEUS luminosity monitor makes use of the bremsstrahlung process $ep \rightarrow ep\gamma$. This is observed using an electron/photon detection system 30m downstream of the beam crossing point in the electron forward direction. The systematic error on the luminosity is expected to be no more than 5%. This system also allows the observation and tagging of electrons scattered at small angles to the beam direction during photoproduction events.

Most HERA collisions result in proton debris travelling along the beampipe. ZEUS is able to study the leading proton in this debris using a system of six roman pots spread over 100m in the proton forward direction.

2.6 Readout and Triggering

The readout and triggering environment at ZEUS is dominated by the short 96ns beam crossing interval and the high rate of background events. Overcoming these problems has presented a major challenge to current detector technology. The experience gained while working within this environment will play a vital role in the design of detectors for the next generation of accelerators, which will have even shorter beam crossing intervals. An outline of the ZEUS online systems is presented in chapter 4.

2.7 Summary

ZEUS is a general purpose experiment, well designed to exploit the physics possibilities of HERA. The construction of an experiment of this scale, by physicists from three continents, is a triumph for the international scientific community.

Bibliography

[1] ZEUS collaboration, letter of intent 'ZEUS - a detector for HERA', DESY June 1985.
ZEUS collaboration, 'The ZEUS detector - a technical proposal', DESY March 1986.
ZEUS collaboration, 'The ZEUS detector - status report 1989', DESY March 1989.

[2] C.W.Fabjan and T.Ludlam, Phys. Rev. Lett. 60B (1975) 105.
C.W.Fabjan et al, Nucl. Instr Meth 141 (1977) 61.
T.Akesson et al, Nucl. Instr Meth A241 (1985) 17.

Chapter 3

The Central Tracking Detector

The ZEUS Central Tracking Detector (CTD) is a large cylindrical drift chamber [1]. In this chapter we shall

- Give an overview of the important operating parameters of the CTD.
- Discuss track finding in this environment.

3.1 Overview

The CTD forms the barrel of the ZEUS charged particle tracking system, presenting 72 sense wire layers to a track traversing its complete active radius. A typical event in the CTD is illustrated in figure 3.1. For tracks perpendicular to the proton beam direction, the expected momentum resolution is $\sigma(p)/p = 0.0021p \oplus 0.0029$. Other important chamber parameters are summarised in table 3.1.

Position resolution	100-120 μm (θ dependant)
z resolution (stereo)	1.2 mm
z resolution (timing)	50 mm
Two-track resolution	1.6 mm
Magnetic Field	1.8 Tesla
Lorentz angle	45°
Maximum drift time	500 ns
Active length	2024 mm
Active radius	190-785 mm

Table 3.1: Summary of CTD parameters.

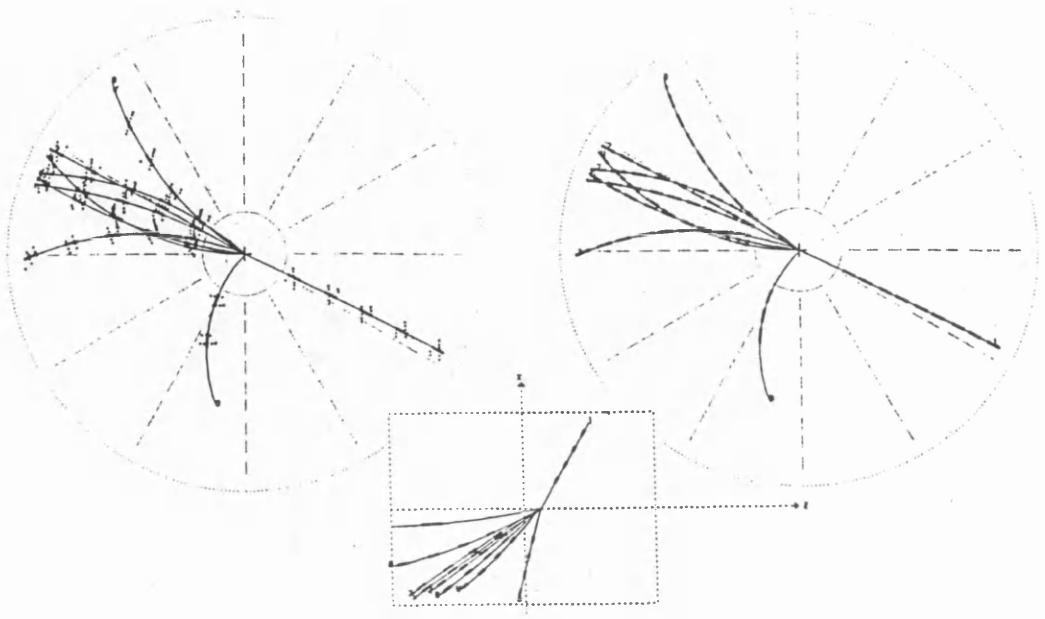

Figure 3.1: A NC event inside the CTD.

Drift chambers reconstruct points in space by measuring the time taken for ionisation deposited by a passing high energy particle to drift to a sense wire. Unfortunately the left-right ambiguity of the drift direction leads to the creation of an additional 'ghost' point associated with each drift time.

The CTD is organised into nine concentric superlayers, each formed by repeating a small drift cell of eight sense wires. This geometry is illustrated in figure 3.2. Inside a cell the sense wires are arranged to lie in a plane inclined at 45° to the radial vector at the centre of the cell. The high Lorentz angle thus ensures that ionisation drift is approximately tangential.

This CTD design forces most tracks traversing a superlayer to cross a geometrical boundary either between or within cells. Only real tracks need be continuous at such boundaries:

1. Left-right ambiguity is resolved locally, within a superlayer, for track segments crossing an inter cell boundary.

2. Out-of-time tracks are rejected locally for track segments crossing an inter cell boundary *or* a sense wire plane.

The odd numbered superlayers are axial superlayers. These contain wires which run parallel to the beam axis, providing hit information in the $r\phi$ plane.

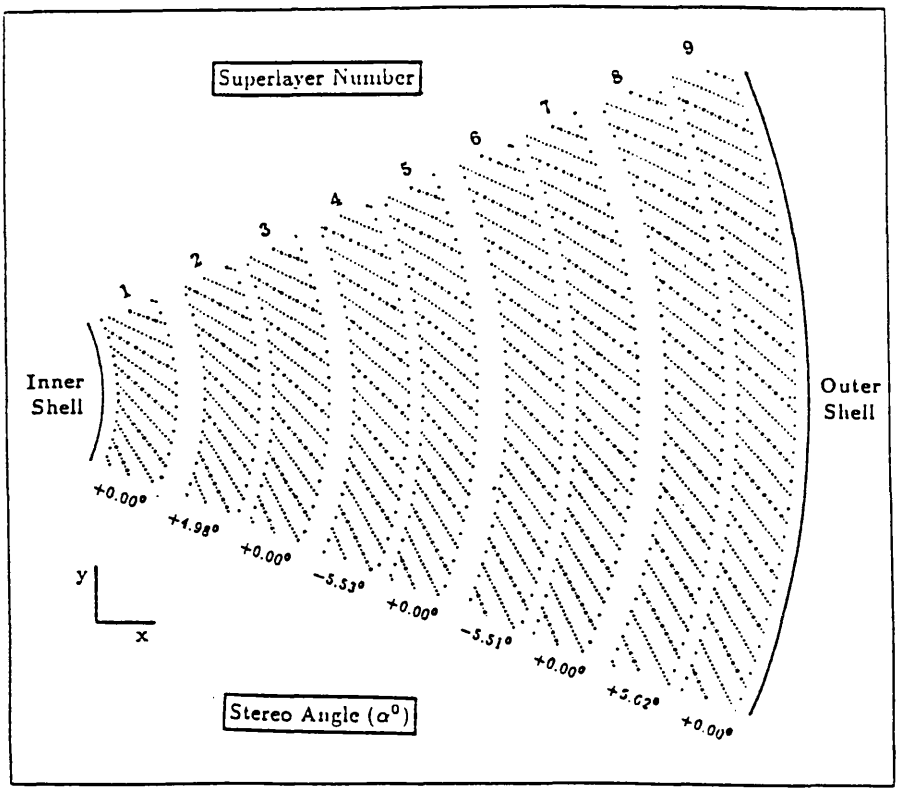

Figure 3.2: The cell and superlayer structure in an azimuthal sector of the CTD.

The alternate superlayers contain stereo wires, rotated by angles of up to $\pm 5°$. These allow tracks to be reconstructed in z.

All sense wires in superlayer 1, and half those in superlayers 3 and 5 are equipped for additional direct z-readout. This is done by measuring the difference in arrival time of the charge pulse at each end of the sense wire. This z-by-timing system is principally for use in the ZEUS first level trigger system.

3.2 Track finding in the CTD

The geometry of the CTD suggests a two stage approach to full track finding:

- Find tracks in 2D using information from the axial superlayers.
- Add the information from the stereo superlayers to form 3D tracks.

This thesis will deal most closely with the first of these operations, investigating its implementation in both the online and offline environments.

When finding 2D tracks within the CTD we face many problems. The large gaps between axial superlayers, and the high rate of hits on wires near the beam pipe are just two complications which we must overcome.

Before the first stages of pattern recognition are complete, we are not able to make the detailed drift time corrections that allow us to access the full position resolution of the chamber. Throughout this thesis we shall assume that the effective position resolution with which we must work is 130μm.

3.2.1 Co-ordinate Systems

[1]We now define two co-ordinate systems suitable for use during track finding in the CTD.

The CTD Co-ordinate System.

The origin of the CTD co-ordinate system is defined to be at the centre of the chamber. The z axis is coincident with the chamber axis, increasing in the same sense as the proton beam direction. The x axis passes through the central ground wire of cell 32 in superlayer 1 and the y axis completes a right handed orthogonal co-ordinate system.

Cylindrical polar co-ordinates are defined for the CTD as follows:

$$\begin{array}{lcll} r^2 & = & x^2+y^2 & 0 \leq r \\ \phi & = & \tan^{-1}(y/x) & 0 \leq \phi < 2\pi \\ \theta & = & \tan^{-1}(r/z) & 0 \leq \theta < \pi \end{array}$$

All distances are measured in centimetres and all angles are measured in radians.

The Local Non-Orthogonal Co-ordinate System.

One local non-orthogonal (LNO) co-ordinate system is associated with each CTD cell.

The u axis is defined to lie in the plane of the sense wires, while the v axis is parallel to the nominal electron drift direction. The uv plane is parallel to the CTD xy plane. The w axis completes a right handed co-ordinate system, as illustrated in figure 3.3, running along the central ground wire of the cell. The vw plane is referred to as the reference surface of the cell. All LNO distances are measured in centimetres.

In stereo superlayers the w axis is not perpendicular to the uv plane, and to further complicate the geometry the u and v axes are rotated as w increases. Fortunately, in the approximation that the magnetic field is constant over the length of a cell, the angle between the u and v axes remains constant. If the electric and magnetic fields take their nominal values this angle is 135°

[1]These definitions were recommended in [2]

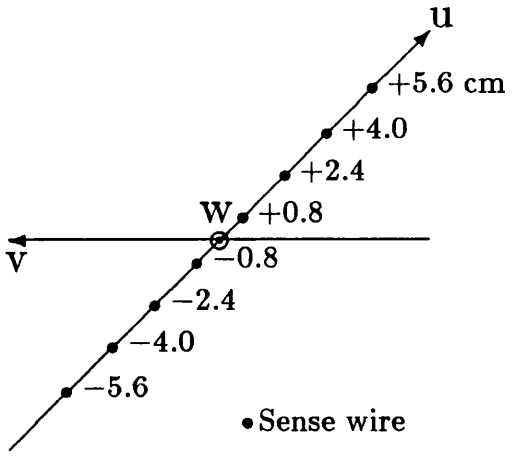

Figure 3.3: LNO co-ordinate system.

3.2.2 Track Elements

The position and direction of a track element can be specified using a helical track model.

A general helix is described by five parameters. However, to uniquely define the position and direction of a helical track element we need to specify six values. To allow the formation of a non-singular covariance matrix we choose to express each track element at the point where it crosses a suitable reference surface. In the CTD it is convenient to use the reference surface of a local cell for this purpose (see section 3.2.1). Track element **P** may then be defined,

$$\mathbf{P} = (\phi, \psi, z, \Theta, Q/P_t)$$

where ϕ and z are the CTD polar co-ordinates of the intersection point, ψ is the angle formed between the projection of the track element on the CTD xy plane and the CTD x axis, Θ is the angle between the track element and the z axis, P_t is the transverse momentum of the track element, and Q indicates the sign of the particle charge.

The following units and ranges are used:

$$\begin{gathered} 0 \leq \phi < 2\pi \text{ rads} \\ 0 \leq \psi < 2\pi \text{ rads} \\ |z| < \infty \text{ cm} \\ 0 \leq \Theta < \pi \text{ rads} \\ |Q/P_t| < \infty \ \ (\text{GeV/c})^{-1} \end{gathered}$$

If B_z is the local axial magnetic field (Tesla) and c is the speed of light (cm/s), then ρ, the inverse of the local radius of curvature of the track ele-

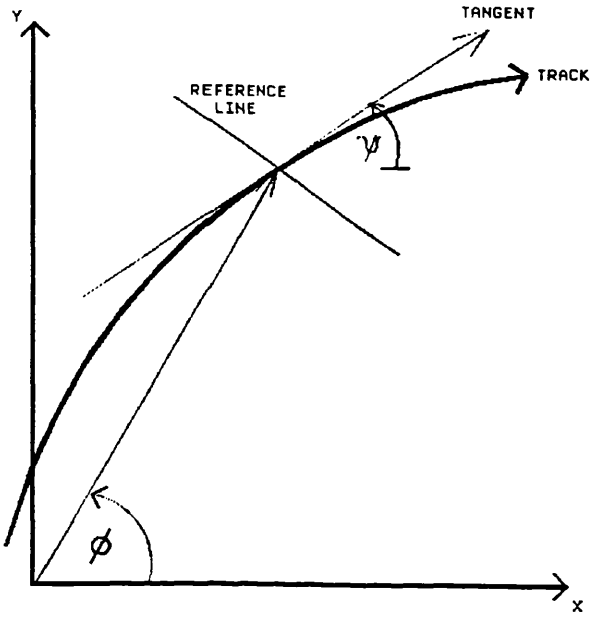

Figure 3.4: Parameters defining a 2D track element.

ment (cm^{-1}), is given by,

$$\rho = B_z \times c \times 10^{-13} \times Q/P_t$$

During two dimensional (2D) pattern recognition helical track elements are projected on to circles in the CTD xy plane. These circular track elements, p are described by only three parameters which must be specified with respect to a reference *line*. The reference line is chosen as the intersection of the reference surface of a suitable cell and the CTD xy plane.

$$\mathbf{p} = (\phi, \psi, Q/P_t)$$

where all variables are defined as before. This is illustrated in figure 3.4.

3.2.3 Analysing Pattern Recognition Performance

[2]The evaluation technique that has been developed for use in this thesis is presented in detail in appendix A. To assess the performance of a track finding algorithm we use simulated 'Monte Carlo' data. We first attempt to match

[2]This methodology was first presented in [3]

each reconstructed track to a Monte Carlo track. Once this *assignment* has been made the quality of the assigned tracks can be assessed.

If all reconstruction was perfect then track assignment would be trivial. However, inefficiencies and the contamination of good tracks with incorrect points leads to problems. One strength of our evaluation technique is that two different approaches are used for track assignment. Each of these provides a different perspective on the performance of the algorithm under study.

After assignment, fiducial cuts may be placed on the sample of Monte Carlo tracks. This ensures that only tracks in which we are interested are counted against efficiency.

We quantify our analysis by dividing Monte Carlo and reconstructed tracks into several intuitive categories: nasty tracks, found tracks, split tracks, and spurious tracks. The quality of a correctly found track is assessed by considering the fraction of true Monte Carlo hits it contains, and the extent to which it is contaminated by hits from other tracks or by ghost points from its own. All of these concepts are defined fully in appendix A.

3.3 Summary

The CTD is a large general purpose drift chamber. Its geometrical features are a great aid to track finding, which can be divided naturally into two stages: finding 2D tracks using information from the axial superlayers, adding 3D information from the stereo superlayers. In this thesis we shall deal primarily with the first of these tasks, and will make extensive use of a specially designed technique for assessing the performance of pattern recognition algorithms.

Bibliography

[1] C.B.Brooks et al, Nucl. Instrum. Methods A283 (1989) 477-483.

[2] K.Long & D.Shaw, 'Geometrical Considerations, Co-ordinate Systems and Co-ordinate Transformations for Event Reconstruction in the CTD' ZEUS-RAL-89-10.

[3] D.Shaw, K.Long & D.Gingrich 'A Technique for Evaluating Pattern Recognition Performance in Tracking Chambers' Oxford preprint OUNP-90-2.

Part II

Online Pattern Recognition

Chapter 4

The Online Environment

The online systems of a high energy physics experiment are those which function continuously while it is recording data. Such systems must read the raw data from each detector channel, compress this information, and eventually gather it all together for output and storage. At various stages in this process it may be necessary to form a *trigger*, deciding to either accept or reject an event before further readout. This can greatly reduce the volume of data flowing through the system as rejected events can be immediately discarded.

To form a trigger decision it is often necessary to use pattern recognition techniques. This could involve track reconstruction in a drift chamber or cluster finding in a calorimeter. Depending on the complexity of the task and the time available, a variety of hardware and software techniques are available. In the context of this thesis we shall discuss the use of modern programmable processors in this role.

This chapter presents the discussion in two parts:

- An overview of the ZEUS trigger system.
- A comparison of three modern processors.

4.1 An Overview of the ZEUS Trigger System

Electron-proton bunch crossings occur within the ZEUS detector every 96ns. To appreciate the implications of this, let us consider the ZEUS inner tracking detectors: 11,844 channels are read out continuously, each analogue signal being digitised by a 100MHz 8-bit 'flash' analogue to digital converter (FADC) system. This produces a raw data rate of almost 1.2 terabytes per second! The online systems face a formidable problem: how to cope with this enormous data rate?

Few bunch crossings produce an event of interest. However, many are associated with background activity. To reduce this background rate while processing such a large amount of data, a multistage trigger architecture is essential: each

Bunch crossing	10 MHz
First level trigger accepts	1 kHz
Second level trigger accepts	100 Hz
Third level trigger accepts	$1 \sim 3Hz$

Table 4.1: Design rates for the ZEUS trigger system.

stage having progressively more time at its disposal to decide which events it should keep and which can be safely discarded. Table 4.1 shows the design rates for each stage of the ZEUS trigger system.

At HERA the most important source of unwanted background events are beam-gas collisions in the upstream proton beam pipe. A high energy proton collides with a residual gas molecule in the beam-line vacuum. This can lead to a large spray of particles, and even if the collision occurs up to 80m away in the proton upstream beam-pipe, tracks may find their way into the ZEUS detector [2].

There are three principle ways that the trigger system can try to discriminate against beam-gas events:

1. Insisting that the event has a large total transverse energy, $\sum |E_t|$. This scalar sum can be measured by the calorimeter.

2. Constraining the primary vertex of the event to be consistent with the nominal interaction region. This uses information from the inner tracking detectors or from special 'veto' scintillators.

3. Searching for high transverse momentum leptons. These might produce: activity in the muon chambers, significant electromagnetic deposits in the calorimeter, or a large missing transverse momentum, $\left|\sum \vec{P}_t\right|$ observed in the calorimeter.

Each stage of the ZEUS trigger system will use some combination of these techniques to achieve the desired trigger rate.

4.1.1 First Level Trigger

The ZEUS first level trigger (FLT) has to make a decision every 96ns, with a maximum delay, or *latency* between the event and the decision of only 5μs[1]. The FLT cannot be formed in software, as even the most powerful programmable processors currently available would not meet these stringent requirements.

[1]This corresponds to the maximum number of events that the hardware is designed to store.

The FLT relies principally on the fast hardware electronics within two sub-detectors: the calorimeter and the CTD. The calorimeter uses the fast response of its scintillator readout to form a $\sum |E_t|$ value. The CTD is able to use its special z-by-timing readout to reconstruct tracks in z using a hardware hit array. This provides information about the possible position of the primary vertex. Further discussion of the ZEUS FLT is beyond the scope of this thesis.

4.1.2 Second Level Trigger

The ZEUS second level trigger (SLT) has to make a decision every 1ms, with a latency of 15ms. In this role the advance of modern computing technology has made the use of programmable processors possible, though the design and implementation of such systems remains a difficult task.

Tracking information will be an important part of the SLT. The CTD SLT track finder should have the following properties,

- Find large P_t tracks with high efficiency.
- Measure 3D track kinematics as well as possible.

This information can be used to determine event characteristics such as the position of the primary vertex and charged track multiplicity of an event. Track trajectories in the CTD can be correlated with information from other SLT components.

In a naive system the CTD SLT would have less than 1ms to complete all its pattern recognition processing. Three design features can ease this problem:

Pipelining: The algorithm is divided into several sequential parts, these may be executed by a series of processors connected together by a *pipeline*.

Geometric parallelism: A large number of processors are used in parallel, executing the same part of the algorithm on data from different parts of the detector.

Buffering: Memory space is provided to *buffer* data between the different processing stages. This means that each stage need only match the 1kHz rate on average, rather than having to complete its task in less than 1ms for each individual event.

These design techniques allow us to add to the processing power of our system and to extend the maximum time available for each processing stage. Only the overall SLT design, requiring that the system have a latency of no more than 15ms, places a restriction on the number of pipelined stages we can employ and the maximum occupancy of our buffers.

Track finding within the CTD SLT is one of the major themes of this thesis, and will be discussed in chapters 5 and 6.

4.1.3 Third Level Trigger

The ZEUS third level trigger (TLT) is the final stage of the ZEUS trigger system. For events accepted by the SLT, data from each sub-detector is gathered together by the ZEUS event builder (EB). Packaged events are then passed to a processor farm which implements the TLT. This shares many features of the offline processing environment, and is discussed fully in chapter 8.

4.2 Modern Microprocessors

To understand what can be achieved within the time constraints of trigger processing, we must first consider the tools at our disposal: a wide range of modern microprocessors. In this section we shall compare three types of processor, each with its own distinctive architecture.

4.2.1 Digital Signal Processors

A digital signal processor (DSP) is a small microprocessor optimised for high speed numeric processing operations.

In 1990 world sales of DSPs are expected to top $375 million, with applications including image processing, radar and sonar, speech processing, and telecommunications [1]. This rapidly expanding market is supplied by a wide range of commercial suppliers, including AT&T, Analog Devices, Motorola, and Texas Instruments.

Most DSP designs are based on either 16-bit or 32-bit internal architectures. The 16-bit designs tend to be the cheaper and faster but are only suited to fixed-point arithmetic. The more complex 32-bit designs have the advantage that they can perform efficient floating-point operations. In 1989-90 fixed-point designs were used by 95% of commercial applications [1].

A general purpose DSP typically contains at least two types of specialised computational units:

- An arithmetic and logic unit, providing condition testing and basic addition and subtraction operations.
- A multiplier unit, providing a single cycle multiply/accumulate instruction.

Each of these would be supported by several independent input and output registers, allowing the flexibility needed for the efficient evaluation of complex numerical expressions.

Many digital signal processing algorithms call for highly repetitive memory access, processed by a small loop of code. To implement such an algorithm in an efficient manner three features must be added to the DSP architecture:

1. Zero overhead looping.

2. Zero overhead address generation.

3. Parallel data and instruction streams (Harvard architecture).

The first of these is achieved via special looping hardware, the second through the use of one or more independent address generators. These normally allow each address to be incremented after use, greatly speeding up access to arrays of numbers. Maximum computational performance can only be sustained if addressing data in memory does not interfere with the ability of the processor to fetch its next instruction. In a conventional *von Neuman* architecture, data and instructions are stored in the same address space and must be fetched serially. To overcome this problem DSPs often use a *Harvard* architecture, with multiple on-chip buses supporting parallel access to two separate address spaces.

Modern high energy physics experiments increasingly make use of FADC units to digitise the analogue output of their detector channels. The amount of data generated in this way can be very large in relation to its information content. DSPs are the natural choice to process this data as the first stage of an online data aquisition system, greatly reducing the amount of data that must be passed on to the rest of the system.

An example of a high performance DSP

The ADSP 2100 is an example of a typical general purpose DSP with a 16-bit architecture. One design feature that makes it different from many other DSPs is that it contains very little on-chip memory. Instead it extends the Harvard architecture off-chip by using separate data and instruction buses to external memory. To summarise its architecture, it has

- An off-chip Harvard architecture.
- Three computational units supported by a large register set.
- Zero overhead address generation.
- Zero overhead conditional looping.
- Single cycle execution of all instructions.
- 80ns cycle time (12.5MHz version: ADSP 2100A).

This DSP is notable because of its exceptionally high degree of operational parallelism. In a single cycle it can perform and sustain the following:

1. Save the last result to memory.

2. Test a loop termination condition.

3. Perform a multiply and accumulate instruction.

4. Fetch an operand for the next operation.

5. Increment all address counters.

6. Fetch the next program instruction from cache memory.

The ADSP 2100 architecture underlies a family of DSPs which offer performance improvements combined with upward code compatability. The ADSP 2100A features a 12.5MHz clock rate, resulting in one instruction being executed every 80ns. The more recent ADSP 2101 and 2102 update the range offering an improved price to performance ratio in some applications.

4.2.2 Transputers

A general process, of which parts may be executed in parallel, can be modelled as a network of 'communicating sequential processes' (CSP). In such a system an arbitrary number of conventional sequential programs cooperate by passing messages between themselves.

The OCCAM programming language is a realisation of the CSP model. It allows the definition of multiple processes, some of which are allowed to take place in parallel. Communication is achieved by passing messages along defined *channels* between the processes.

Algorithms with a high degree of parallelism tend to be complex and are subject to two unique problems:

Deadlock: The system never communicates with the outside world because all internal processes are waiting to communicate with each other.

Livelock: The system never communicates with the outside world because all internal processes are indefinitely busy communicating between themselves.

OCCAM tries to reduce these difficulties by providing the facility to program in a high level language, while keeping the syntax compact enough to allow the use of formal methods to verify algorithms.

The transputer is a single chip microcomputer, providing locally a CPU, memory and links for connection to other transputers. It is designed for the efficient execution of algorithms using the processing model outlined above. Algorithms written in OCCAM may be executed on a single transputer or mapped on to a network of transputers to allow truly parallel execution.

The transputer is a 'reduced instruction set computer' or RISC device. In such an architecture the limited number of instructions that the processor can

perform are optimised and chosen to correspond to those most often used in the intended applications. This leads to simplifications in chip design and an improved price to performance ratio.

There are three series of devices within the transputer family:

T200: Architecture is 16-bit, supporting only fixed point arithmetic in hardware.

T400: Architecture is 32-bit, supporting only fixed point arithmetic in hardware.

T800: Architecture is 32-bit, a dedicated computational unit supports floating point operations.

Each of these transputers has up to 4 kbytes of on-chip memory and four bidirectional serial links suitable for connecting to any other transputer. Upward code compatability is retained throughout the transputer range, and any of these processors can be mixed freely in a transputer network.

The best models within each transputer generation provide serial link data rates of 20Mbits/s and a cycle time of 33ns. However, it is a mistake to assess processor performance solely on the basis of cycle time: the T400 requires up to 240 processor cycles to complete a 32-bit floating point multiplication, while this is cut to 11 cycles by the T800 floating point unit.

The transputer CPU uses only six internal registers. These must be shared by all basic operations: performing arithmetic, evaluating logical conditions, calculating data addresses and controlling program loops. This is in stark contrast to the DSP architectures we discussed earlier. These usually provide several registers associated with each computational unit, normally supported by special addressing and looping hardware. This allows the DSP programmer to make great efficiency gains by the careful hand coding of an application. The transputer is not designed for this role, by assumption the parallel algorithms for which it is intended are too complicated for this treatment.

The transputer's great strength lies in the ease with which it can implement complex parallel applications, and in the inherent expandability of any transputer network.

4.2.3 The Intel i860

The i860 from Intel is one of the most powerful microprocessors presently available. Based on a 64-bit design and using RISC techniques, it features

- An on-chip Harvard architecture, accessing separate data and instruction cache memories.
- Three computational units supported by a large register set.

- Automatic incrementation of address pointers.
- Low overhead looping.
- Pipelined execution for instructions taking more than one cycle.
- 30ns cycle time.

These features combine to allow the i860 to support efficient vector operations: 64-bit numbers from the data cache can be continually input into the pipelined floating point unit, which can perform parallel multiply/add operations producing one result per clock cycle (see chapter 8).

The architecture of the i860 has been presented in such a way as to invite comparison with the DSP design presented previously. It is important to realise that despite their apparent similarities, these two devices offer very different challenges to the programmer. The DSP has a relatively simple instruction set, allowing applications to be coded and optimised directly. In contrast, the pipelined structure of the i860 means that it is difficult to program it in this manner. Access to the power of this processor is most easily achieved by using a high level language and an efficient vectorising compiler.

4.3 Summary

This chapter has given an overview of the online processing environment, with particular reference to the ZEUS detector. In the following chapters we shall see each of the design techniques and microprocessors we have discussed come to play a part in the structure of the CTD SLT and potential applications at future accelerators.

Bibliography

[1] Computer Design, April 1 1990, pg 82.

[2] D.Gingrich, Nuclear Physics Lab., Oxford, Private communications, Sept. 1990.

Chapter 5

Preliminary Online Study

[1]The design of the CTD SLT is crucially influenced by the processing power required for each type of operation it might have to perform. Only when we understand the relative requirements of some basic types of operations can we begin to design the final system. To begin our investigations we shall consider the three basic components required for track finding within this environment:

1. Finding hits from the raw chamber output.
2. Reconstructing track segments within a superlayer.
3. Matching segments between axial superlayers to form 2D tracks.

To produce timing estimates for these operations we use as reference the ADSP 2100 (see chapter 4). Using the experience gained with this processor we are able to predict the likely performance of other DSP's and Transputers.

5.1 Finding Hits

5.1.1 Signal and Noise

In order to perform its role, the CTD SLT must be closely integrated into the CTD readout electronics. The analogue signal from each sense wire is taken to a readout card, where it is digitised by a 100MHz 8-bit FADC system into 10ns bins. The maximum drift time within the CTD is 500ns, and so for each wire a train of fifty bins is associated with a CTD event.

The signal that we wish to extract from this data is in the form of a *pulse*. Figure 5.1 shows the characteristic shape of such a pulse: a sharp leading edge followed by a longer falling tail. The height and width of the leading edge can be used to distinguish a genuine pulse from background noise. In addition, it is

[1]This work was first presented in [1, 2, 3]

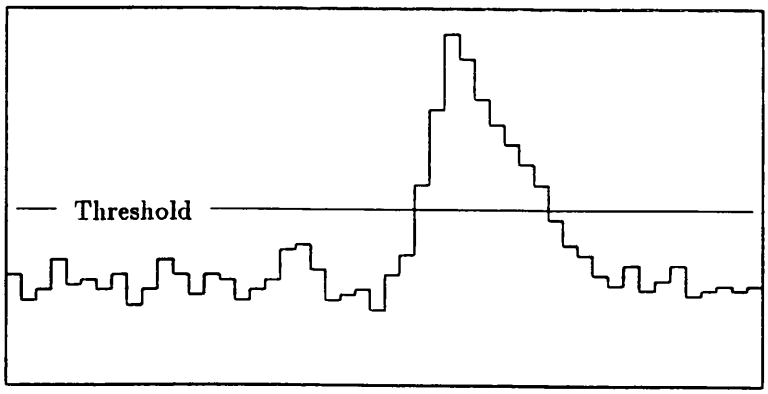

Figure 5.1: A typical digital pulse.

convenient to introduce the concept of a *threshold* level above which background noise seldom fluctuates.

To reduce the amount of data that must pass through the online system, it is important that we find and parameterise pulses at an early stage. We must measure two important features of each pulse:

Drift time: The arrival time of the leading edge of the pulse allows us to measure the position of the passing track. This is the principle of operation of any drift chamber.

Integrated area: The integrated pulse area is a measure of the total amount of ionisation deposited by the passing track. This varies with particle type and energy, and forms the basis for particle identification by the dE/dx method.

When a pulse has been parameterised in this way it is known as a *hit*. Hit finding is the first stage of programmable processing carried out by the CTD online system, it forms the basic input to the CTD SLT.

5.1.2 Simulation Timings

A prototype hit finding algorithm has been implemented on an ADSP 2100 software simulator. This allows an accurate measurement of the processing time required by this basic operation.

The algorithm is a variation of the 'constant fraction discriminator' (CFD) technique [4]. We consider a pulse train 50 bins long, where bin I has a value A, and proceed thus:

1. Check a hit-flag indicating whether this channel might contain a pulse.

	μs
Check hit-flag	0.5
Scan 50 bins	1.3
Apply cut on width	1.9
Apply cut on height	2.6
Calculate drift time	2.5
Integrate area	2.2

Table 5.1: Timings for basic hit finding operations.

2. Search through the channel, checking every third bin to see if it is above the threshold level (skipping bins increases speed).

3. When such a bin is found, search backwards to find the last bin that was below the threshold level, this bin is called the *pit*.

4. Starting at the pit, search forwards until the signal no longer rises, this bin is called the *peak*.

5. Make an acceptance cut on the width, $I_{Peak} - I_{Pit}$.

6. Make an acceptance cut on the height, $A_{Peak} - A_{Pit}$.

7. If the pulse passes these tests then it is accepted as a hit and we calculate the CFD level $= A_{Pit} + (A_{Peak} - A_{Pit})/4$

8. We calculate the drift time by extrapolating between the bins I and $I+1$ on either side of the CFD level.
   Drift time $= [I + (CFD - A_I)/(A_{I+1} - A_I)] \times 10$ns

9. The area of the pulse above the CFD level is integrated.

10. To improve the two hit resolution, a special routine is used to search the tail of the pulse for potential second hits.

This software has been optimised exclusively for speed of execution.

- ADSP 2100 assembly language has been used throughout.
- All calculations use fixed-point arithmetic.
- A simple algorithm is employed, specially adapted to the ADSP 2100 architecture.

		μs
Hit pulses	=	$16 \times 0.2 \times 11.0$
		35.2
Noise pulse	=	$16 \times 0.2 \times 6.3$
		20.2
Empty channels	=	$16 \times 0.6 \times 0.5$
		4.8
Total time		60.2

Table 5.2: A summary of hit finding for 16 channels.

The timing results are presented in table 5.1. The figures quoted are independent, and so the total time for a given channel is the sum of the operations used. For example, a channel containing just one good pulse would require processing for 11.0μs to produce a fully parameterised hit, while a channel containing this and an additional noise pulse (rejected on a width cut) would take 12.9μs

Consider an architecture where one DSP deals with 16 channels (in chapter 6 we shall see that this is a convenient choice). Most of the events that this system would have to process would be beam-gas events. In [5] it was estimated that even in superlayer one an average of only 5% of wires will be hit in such an event. Let us assume that this figure increases to 20%. In addition to this we assume that 20% of channels contain wide noise pulses above threshold, the rest being empty. The implications of this analysis are summarised in table 5.2. This simple calculation assumes that all buffers are infinitely deep so that fluctuations are unimportant.

We conclude that a high performance DSP can perform these basic functions with a wide safety margin and still have over 900μs left to spend on other tasks. These might include helping to find track segments, or implementing a more complex and more efficient hit finding algorithm.

5.2 Hits $\rightarrow$ Segments

The superlayer structure of the CTD is well suited to a two stage track finding process:

- Find track segments within each superlayer.
- Match segments between axial superlayers to form 2D tracks.

This type of algorithm also has the advantage that it can be pipelined and can make extensive use of geometric parallelism (see chapter 4). It is expected that

the first of these operations, track segment reconstruction, will be the most time critical. In this section we shall investigate the implementation of this operation.

The region of the chamber currently being searched for segments is called the pattern recognition *mask*. Within the CTD, a single cell is the simplest possible mask, and conveniently allows us to reconstruct segments using the LNO coordinate system (see section 3.2.1). To achieve the maximum performance we choose to redefine the scale on the LNO u axis to be *wire units*, running from -3 to $+4$ between the inner and outer sense wires respectively. The LNO v axis can also be scaled if required. As we shall see, these alterations aid the speed and numerical precision of our algorithms.

Unfortunately, the single cell mask is limited in the scope of reconstruction it allows. A typical track entering a CTD superlayer leaves its hits spread between two adjacent cells, and so to find track segments with the maximum number of hits we must use a multiple cell mask. As a further advantage, this would allow the direct rejection of segment left-right ambiguity and out of time tracks (see chapter 3). We will proceed on the assumption that the increased speed of the single cell mask is more important than its inferior pattern recognition potential.

The segment reconstruction operation can be divided into two stages:

Segment finding: Decide which points belong to a segment.

Segment fitting: Perform a straight line fit to this list of points.

5.2.1 Segment Finding

The choice of SLT segment finding algorithm is critically constrained by the short processing time available. We will investigate the suitability of each of the following algorithms:

1. Road method.
2. Cluster finding in m.
3. Cassel-Kowalski tree algorithm.
4. Segment following.
5. Template method.
6. Minimal segment finding.

The following timing estimates are based on an outline of how each algorithm might be implemented on the ADSP 2100. The results are quoted in terms of machine cycles, and it is expected that these are accurate to about one cycle

Operation	Machine cycles		μs
Choose pair and form road	4	=	0.32
Check one hit from hit list	10	=	0.80
Iterate for all hits	$\times 24$		
Iterate for all possible roads	$\times 448$		
	Total time	=	∞

Table 5.3: The road method.

per item. All looping and address generation overheads are neglected: this is realistic for the ADSP 2100 because of its special hardware (see chapter 4).

This procedure is suitable for comparing the different algorithms, but cannot be expected to give as accurate an overall result as direct simulation. The relative timing between any two algorithms can be used to estimate their relative performance on any similar DSP or Transputer.

In each of the following discussions we make the general assumption that the pattern recognition mask is crossed by two tracks, each leaving eight hits, and thus contains a total of 32 left-right ambiguous points.

Road Method

This method begins by choosing any two hit points and joining them with a straight line. We now check each other hit point in the pattern recognition mask to see if it lies on this line. There is a subtlety here: we perform each of these calculations in integer arithmetic, the finite precision of which produces a tolerance, or *road* around the line. We may tune the width of our road by changing the scale on the LNO v axis. A segment can be formed from a road containing a suitable number of hit points. This procedure is repeated for each pair of initial hit points. Table 5.3 gives estimated timings for this method.

It is clear that this method could be used, but only if some suitable way of restricting our choice of roads was found.

Cluster Finding in m

We join each pair of hit points with a straight line, $y = mx + c$. The lines associated with a segment form a cluster in m, c space. For the SLT we must try to find these clusters in the fastest possible way: here we consider forming a one dimensional histogram in the gradient parameter, $m = (y_1 - y_2)/(x_1 - x_2)$. In all cases the denominator must be pre-calculated, inverted and stored to avoid a time consuming division operation. To allow rapid access to the histogram bins, we arrange to use each m value directly as an address in memory. As we

Operation	Machine cycles		μs
Calculate m value	7	=	0.56
Increment bin counter	4	=	0.32
Store pair on a list	4	=	0.32
Iterate for all pairs	$\times 448$		
Find maximum from 100 bins	100×2		16.00
	Total time	=	553.60

Table 5.4: Cluster finding in m (all possible pairs of hit points).

Operation	Machine cycles		μs
Calculate m value	4	=	0.32
Increment bin counter	4	=	0.32
Store pair on a list	4	=	0.32
Iterate for all pairs	$\times 112$		
Find maximum from 100 bins	100×2		16.00
	Total time	=	123.52

Table 5.5: Cluster finding in m (pairs of hit points one wire layer apart).

again use integer arithmetic, we can tune the tolerance of our histogram bins by changing the scale of the LNO v axis.

Let us assume that we choose a histogram size of 100 bins. Table 5.4 presents timing estimates for this technique using all possible pairs of points, while table 5.5 uses only pairs of points one wire layer apart. Notice that in the latter case not only is the total number of combinations reduced but also the calculation of each m value is simplified.

We see from these tables that this method can be made to run at high speed. However, using a histogram in only one dimension would presumably limit the pattern recognition performance.

Cassel-Kowalski Tree Algorithm

A *link* is constructed between each pair of hit points. Two links are joined to form an *elementary tree* if they are consistent with coming from the same segment. Segments are reconstructed by joining these elementary trees together in a recursive *climbing* process [6].

The logical structure of this method makes extensive use of arrays and pointers. If we only use links between hits one or two layers apart, then in our standard mask there are 208 possible links. If we reserve 6 words in memory for each link, we will need a total of 2.5 kbytes to complete our data structure. Even if it were possible to fill this large data structure at the rate of one word

Operation	Machine cycles		μs
Calculate current m value	4	=	0.32
Predict hit on next wire	4	=	0.32
Check one hit against prediction	10	=	0.80
Average number of checks required per wire	$\times 1.5$		
Iterate for predictions to all wires	$\times 6$		
Average number of iterations to find all tracks	$\times 2.5$		
	Total time	=	32.40

Table 5.6: Segment following.

every few machine cycles, it would still take an unacceptably large amount of time. Only after this preparatory stage could the relatively rapid tree climbing process begin.

Segment Following

Segment following combines some of the better qualities from each of the previous methods. This algorithm has the advantage that it works with drift times rather than left-right ambiguous points [7].

Starting with a pair of hits one wire layer apart, the difference in their drift times is calculated. This is used to predict the drift time of a hit on the next wire in the cell. Each hit on this wire is then tested in turn against this prediction. If a hit is found then it is used along with the hit on the previous wire to make another prediction. This process is repeated for all wires. We are free to allow missing hits along segments, and to insist that no hit is used more than once.

Estimated timings for this algorithm are given in table 5.6. These are very encouraging, and as we shall see this method will form the basis of our preferred segment finding strategy.

Minimal Segment Finding

When evaluating the performance of different segment finding methods, it is instructive to compare them to a simple or *minimal* strategy.

The hits from all singly hit wires in the mask are placed on to one segment. In doing this we must decide which sense of the left-right ambiguity should be used for each hit. This decision is made by observing the difference in drift times between successive hits, and noting when this value changes sign. Though this procedure is inexact, it is very fast.

This algorithm fails totally on the standard mask we have been discussing, but will return to provide a useful point of reference in the next section and

later chapters.

5.2.2 Segment Fitting

When a track with $P_t = 1$ GeV/c traverses a superlayer, it deviates from a straight line by only 1.3mm. It is often convenient to represent such a 2D track segment as a short straight line within the superlayer. This *vector* hit has a gradient that is approximately tangential to the true track trajectory in this region.

The equation of a straight line in the LNO co-ordinate system is

$$v = mu + c$$

Using square brackets to denote a sum over n hit points (u_i, v_i) we define Δ^2

$$\Delta^2 = [c + mu_i - y_i]$$

If we minimise Δ^2 with respect to the parameters m and c, we have a least squares fit:

$$m = \frac{[u_i v_i] - [u_i][v_i]}{n[u_i^2] - [u_i]^2} \qquad c = \frac{[u_i^2][v_i] - [u_i][u_i v_i]}{n[u_i^2] - [u_i]^2}$$

The error matrix is given by

$$\frac{1}{n(n[u_i^2] - [u_i]^2)} \begin{pmatrix} [u_i^2] & -[u_i] \\ -[u_i] & n \end{pmatrix}$$

For SLT processing we have made use of Δ^2 because it can be calculated more quickly than a conventional χ^2 value.

Our use of the LNO co-ordinate system is important: it allows us to assume that all hits are expressed at integer values of 'wire units' along the u axis. For a given fit this means that many terms can be calculated in advance: $[u_i]$, $[u_i^2]$, an acceptance cut on Δ^2. In addition the denominators, $n[u_i^2] - [u_i]^2$ and $n(n[u_i^2] - [u_i]^2)$ can be calculated *and* inverted in advance, avoiding time consuming division operations. As there are only about 200 potentially interesting combinations of u values, the amount of memory required by this structure is only $\sim$ 2.4 kbytes.

A prototype version of this algorithm has been implemented in assembly language on an ADSP 2100 simulator. The minimal segment finding algorithm (see previous section) has been used to provide the list of hits to be fitted. The timing results are presented in table 5.7. These correspond to fitting a straight line to a segment of eight hit points.

When implementing this fit on the ADSP 2100 it was critical to ensure that the numerical terms involved did not overflow its 16-bit internal registers. This was only possible because of our special choice of axis scale in the LNO co-ordinate system. A full demonstration that this is possible is given in [2].

	μs
Minimal segment finding	3.8
Basic least squares fit	5.8
Calculation of Δ^2	5.1

Table 5.7: Timings for basic segment fitting operations.

5.3 Segments → Tracks

Once we have reconstructed as many segments as possible, we find 2D tracks by matching their corresponding vector hits between axial superlayers. Here we shall consider only 2D tracks, found from among the track segments in axial superlayers.

In the CTD co-ordinate system, a vector hit is specified by r and ϕ, the polar co-ordinates of its centre, and ψ, its direction in the xy plane (cf. chapter 3). Consider a high momentum track which passes through the origin of co-ordinates at an angle ψ_0, and has a radius of curvature R_0. For each vector hit we define $\chi = \phi - \psi$, which allows a simple representation of the track parameters [8]:

$$r = 2R_0 sin\chi \sim 2R_0\chi \tag{5.1}$$

$$\psi_0 = \phi + \chi \tag{5.2}$$

Our prototype segment matching algorithm makes use of this representation.

The sign of the χ value of a vector hit indicates the charge of its parent track, while within a given superlayer the absolute value of χ is inversely proportional to P_t. We can make use of this natural ordering during track searches by sorting our vector hits into order of increasing χ.

Vector hits contain a higher quality of information than normal hits, and so we anticipate that the pattern recognition required in this stage of track finding will be less time critical than that needed in segment finding. We shall therefore leave further discussion of our segment matching algorithm until chapter 6, where we shall see that this assumption is indeed justified.

5.4 Conclusions

From this chapter we can draw three conclusions:

- Basic hit finding can be comfortably achieved using one high performance DSP per 16 readout channels.
- Of the segment finding algorithms which we have considered the segment following method offers the fastest execution speeds.

- It is expected that segment fitting would take a relatively small amount of time compared to segment finding.

The fast execution speed expected for the segment following algorithm encourages us to believe that it would be possible to implement this technique on a network of transputers within the SLT environment.

In the next chapter we shall show that this is indeed possible, and that this algorithm offers both the execution speed and pattern recognition performance necessary for the task.

Bibliography

[1] D. Shaw. ZEUS-UCL-88-0004, The ADSP 2100, a digital signal processor for ZEUS.

[2] D.Shaw. ZEUS-UCL-88-0005. Online Segment Fitting Using DSP's.

[3] D.Shaw. ZEUS-UCL-88-0007. Preliminary discussion of possible segment finding algorithms for second level trigger processors.

[4] H.Fawcett, York Town Univ., Toronto, Canada. Private communications, May. 1988.

[5] D.Gingrich 'Hit Distributions in CTD FADC Crates' Nuclear Physics Lab., Oxford, Oct. 1988.

[6] D.G.Cassel, H.Kowalski. DESY 80/107. Pattern Recognition in Layered Track Chambers Using a Tree Algorithm.

[7] J.B.Lane, Dept. of Physics and Astronomy, Univ. College London. Private communications, Oct 1988.

[8] P.Schildt et al, NIM 178 571 1980.

Chapter 6

Online Prototype Study

[1]We can now design and test a prototype CTD SLT system. In this chapter we shall

- Describe the prototype architecture of the CTD SLT.
- Investigate our segment finding strategy:
  - Description of algorithm.
  - Performance trials.
- Investigate our segment matching strategy:
  - Description of algorithm.
  - Performance trials.

6.1 Prototype SLT architecture

To place in context the track finding algorithms used in this study we must make assumptions about the processor architecture of the CTD SLT. In this section we will present the design assumed by this prototype study.

The analogue signal from each sense wire is taken to a readout card (ROC), where it is digitised by a 100MHz 8-bit FADC system. Let us assume that each ROC contains 16 such channels corresponding to two CTD cells. The 18 ROCs corresponding to one $\pi/8$ azimuthal sector of the CTD are grouped together in a single electronics crate, 16 of which provide coverage for the entire CTD. On a FLT accept signal the FADC data on each ROC is stored in a buffer. Hit finding is performed by one DSP working on each ROC.

One possible design for the SLT track finding system is illustrated in figure 6.1[2]. Each box represents one transputer, and illustrates the tasks which it

[1]This work was first presented in [1, 2]

[2]This is one of those favoured most recently [3].

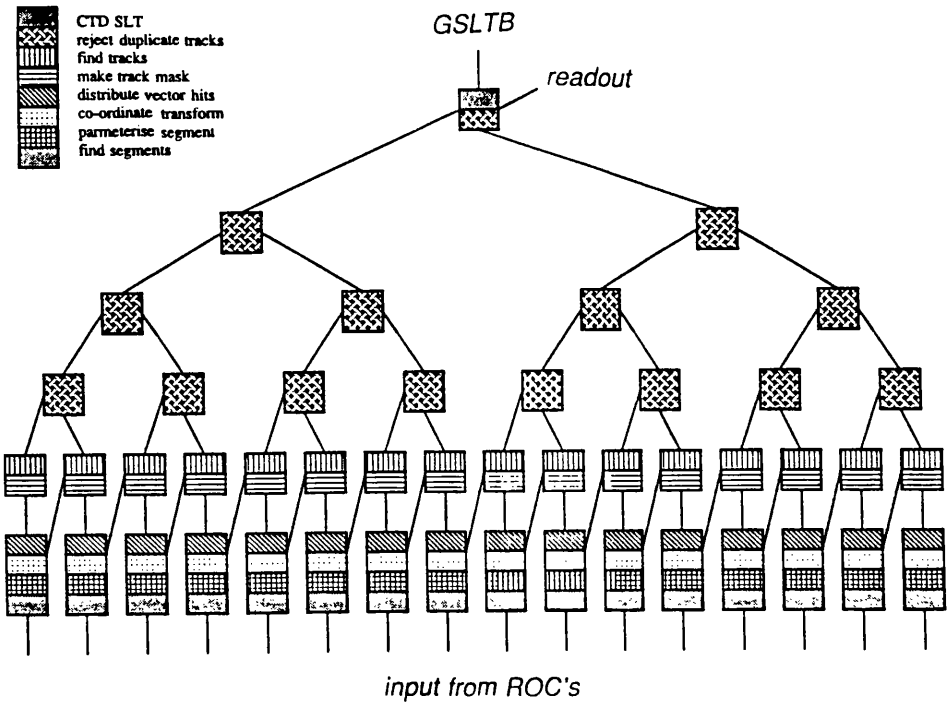

Figure 6.1: One possible SLT architecture.

must perform. At the base of this tree network we find one transputer corresponding to each CTD crate. This implements segment reconstruction for all the single cell masks within that crate and so processing may begin as soon as hit data arrives from each ROC. Segments are matched to form tracks by a second series of transputers. These are able to access the vector hits in both their own and one neighboring crate, thus using a pattern recognition mask equal to a $\pi/4$ azimuthal sector of the CTD. Together these masks overlap to provide complete coverage of the CTD for all tracks with $P_t > 300$ MeV. The final part of the network rejects any duplicate tracks, and brings the remaining information together. The last transputer forms the CTD SLT decision, and passes a summary of the tracking information to both the global SLT board (GSLTB) and the CTD readout system.

The structure of this design makes full use of the great flexibility of a transputer system. This has enabled us to outline a complicated parallel algorithm with the confidence that it will be easily realisable in OCCAM. If at any future time we find that extra processing power is required, it is relatively easy to add extra transputers to such a network.

6.2 Procedure

Using the LEPTO event generator (see appendix B), 1000 NC and 1000 CC events were generated with $Q^2 > 100$ GeV^2.

Beam-gas background events were simulated using the FRITIOF event generator [4]. This provided 2000 pp interactions, uniformly distributed between $-9\text{m} < z < +1\text{m}$ along the proton beam line.

Detector response was simulated using the ZEUS trigger Monte Carlo (see appendix B). This included the position resolution and two hit resolution of the CTD, inefficiencies, noise hits, a realistic magnetic field, and all physics processes such as multiple scattering and energy loss.

In addition the ZEUS FLT was simulated. The trigger algorithm used demanded that the CAL-FLT find $\sum |E_t| > 10$ GeV, *and* that the CTD-FLT find at least one track consistent with coming from the nominal interaction point. Only events satisfying this trigger condition were used in our study. This included 99% of the NC events, 93% of the CC events, and a total of 93 BG events.

These events have been processed within the ARAXNE framework (see appendix C). This allows our algorithms to be implemented within an accurate reflection of our prototype SLT architecture.

6.3 Trial of Segment Finding Performance

6.3.1 Algorithms

The following segment finding algorithms have been implemented. In all cases a single cell pattern recognition mask is used.

Segment following This is our candidate algorithm for SLT segment finding, it has already been described in chapter 5. Segments were required to contain four or more hits, and to have gaps of no more than one hit due to inefficiencies.

Minimal segment finding When evaluating the efficiency of our main segment finding algorithm it is useful to compare it to the performance of a simpler strategy. This minimal algorithm uses less processing time than its sophisticated rival, but offers inferior pattern recognition performance. This algorithm was also described in chapter 5.

Truth segment finding To assess the geometrical acceptance of a particular segment finding technique, it is useful to compare its performance to that of a *truth* segment finding algorithm. This uses Monte Carlo information to achieve the best possible pattern recognition within a given mask.

(%)	NC	CC	BG
Good	34	34	2
Nasty	66	66	98

Table 6.1: Percentage of Monte Carlo segments falling within the fiducial region.

As previously noted, each segment found within a single cell mask is associated with two left-right ambiguous solutions. In our present SLT system we resolve this ambiguity by selecting the solution that forms the smaller angle with the radial vector at the centre of the segment. Due to the high inclination of CTD cells, this technique is expected to work well for segments from high P_t tracks.

Segment fitting is implemented by simply joining a line between the segment end points. Eventually the use of a more sophisticated fitting procedure will allow a small improvement in the final quality of reconstructed segments.

6.3.2 Quality and Efficiency Results

The performance of the segment finder has been assessed using the point matching analysis techniques described in appendix A. These codify some of the intuitive ways in which we assess pattern recognition performance. The following fiducial cuts are used to define the Monte Carlo tracks whose segments we wished to find.

1. $P_t > 1.0$ GeV/c
2. $19° < \theta < 161°$

The first cut excludes low momentum tracks that are of little interest to the trigger, while the second excludes tracks not reaching the middle of superlayer 3. Table 6.1 shows the fraction of Monte Carlo segments from each data set which lie within the fiducial region. For our NC data sample this region always contained the current electron. The figures quoted in this section have statistical errors to within their last digit.

Table 6.2 shows the percentage of good segments which are found by each algorithm for the NC and CC data sets. Approximately 10% of the segments were classified as spurious while only 0.1% of segments were split within a mask. Though the performance of the minimal approach is surprisingly high, the segment following algorithm offers a significantly better performance. The

(%)	NC	CC
Normal	92	93
Minimal	50	32
Truth	99.7	99.9

Table 6.2: Segment finding efficiency.

(%)	NC	CC
Normal	99	92
Minimal	84	66
Truth	99	92

Table 6.3: Acceptance of nominal trigger algorithm using truth segment matching.

very high performance of the truth segment finder indicates that our simple method for selecting the sense of the segment left-right ambiguity is perfectly adequate in this fiducial region.

6.3.3 Nominal Trigger Performance

The segment finding efficiency can be placed in context using a 'truth' track finder. The truth segment matcher always finds a Monte Carlo track if the segment finder produces segments corresponding to that track in superlayers 1 and 3.

Table 6.3 shows the acceptance of a nominal trigger using each of the segment finding algorithms. This simple trigger requires that at least one truth track is found in the event.

The truth segment finding algorithm indicates that using our present method, the best we can do in the CTD for CC events is approximately 92%. For both NC and CC events the acceptance of the segment following and truth segment finding algorithms is virtually identical.

6.3.4 Timing Results

Timing studies were performed on a T800 transputer using our standard FORTRAN implementation of the segment following algorithm (see appendix C). Segment finding was only attempted in masks containing more than 3 and less than 26 hits. Table 6.4 shows the mean time taken by the segment finder to process all of the masks that would be found within a typical CTD readout crate.

(ms)	Time per crate
NC	3.78
CC	4.50
BG	3.77

Table 6.4: Mean time for segment finding on a T800 transputer.

In our prototype architecture this operation is assigned to a single transputer. These times are accurate to within 0.06ms.

Most of the events that the SLT will have to process will be of the BG type. For these the average time taken to complete all segment finding within one crate cannot be longer then 1ms or it will result in a dead-time in the SLT acceptance.

We can see that this implementation of the algorithm seems to be slightly too slow. However, these figures should be viewed as an upper limit, as a substantial increase in speed is expected when the application is coded in OCCAM. Informal studies indicate that an increase by a factor of four is possible [5][3].

6.4 Trial of Segment Matching Performance

6.4.1 Algorithm

The 2D segment matching algorithm uses the vector hit representation defined in chapter 5. Within a superlayer vector hits are stored in order of increasing χ.

Starting with the outermost vector hit in the chamber, we define a road based on its ψ_0 and χ values. Within each superlayer, we note the first vector hit to fall within this road. If the distribution of these vector hits is acceptable as a track, then they are removed from further searches, otherwise only the outermost vector hit is removed. In either case the search is repeated until all vector hits have been accounted for.

In this present study, tracks were accepted if they contained at least two vector hits, one of which had to come from superlayer 1.

6.4.2 Quality and Efficiency Results

The performance of the segment matching algorithm has been assessed using the analysis concepts described in appendix A. The following fiducial cuts were used:

[3]Since this study was completed, this increase in performance has indeed been achieved

Tracks (%)	NC	CC
Found	82.5	57.0
Split	12.4	3.6
Spurious	10.9	17.9
With ghost entries	13.0	13.4
With wrong entries	14.2	26.9

Table 6.5: Pattern recognition performance of prototype SLT track finder.

(%)	NC	CC	BG
Full prototype	99.4%	89.1%	15.0%

Table 6.6: Acceptance of nominal trigger algorithm using the CTD SLT track finder.

1. $P_t > 0.4$ GeV/c

2. $19° < \theta < 161°$

The figures quoted in this section have statistical errors to within their last digit.

Table 6.5 presents the basic pattern recognition performance of the prototype CTD SLT system for our standard NC and CC data sets. This table has been prepared using the point matching analysis technique. Remembering that this is only the first prototype of the segment matching algorithm, these results are encouraging. The track finding efficiencies of 82.5% for NC events and 57.0% for CC events are a firm base upon which we can build.

6.4.3 Nominal Trigger Performance

To place the pattern recognition performance of our prototype system in context, we must assess the trigger efficiency to which it might lead. Following section 6.3.3, we define the acceptance of a nominal trigger: an event is accepted if we reconstruct at least one track with $P_t > 1$ GeV/c. Table 6.6 shows the acceptance rates achieved by our full prototype system. It is reassuring to see that the acceptance rates for NC and CC events is close to that obtained using the truth segments and the truth segment matcher (see table 6.3). This is achieved while reducing the BG background rate by a substantial fraction.

6.4.4 Timing Results

Timing studies were performed on a T800 transputer, using a FORTRAN implementation of the segment matching algorithm.

The time required to find one track was $\sim$ 0.3ms. It was argued in section 6.3.4 that this time would decrease if the algorithm was coded in OCCAM. Even without any further improvement this timing result is very encouraging.

6.5 Conclusions

The results in this chapter can be summarised in two important points:

- The segment following algorithm can be used successfully within a single cell mask. This simple method combines good pattern recognition performance with very fast execution times.
- The performance of the overall track finder is encouraging, offering a nominal trigger acceptance close to the best possible.

We can confidently proceed to design the final CTD SLT system based on these algorithms. Implemented on a network of transputers, matched closely to the sector structure of the CTD, this architecture should provide a flexible framework with sufficient processing power for all our needs.

Bibliography

[1] D.Gingrich, D.Shaw & J.B.Lane 'Segment Finding in the CTD Second Level Trigger' Nuclear Physics Lab., Oxford, Nov. 1989.

[2] S.Smith 'Track Finding in the CTD Second Level Trigger' Nuclear Physics Lab., Oxford, Feb. 1990.

[3] D.Gingrich, Nuclear Physics Lab., Oxford, Private communications, Sept. 1990.

[4] B.Nilsson-Almqvist, E.Stenlund, Univ. of Lund, Sweden. FRITIOF version 1.6, write-up (unpublished).

[5] J.B.Lane, Dept. of Physics and Astronomy, Univ. College London, Private communications, Sept. 1988.

Chapter 7

Applications at Future Accelerators

[1]The next generation of hadron colliders (LHC and SSC) will produce as many as 100 million collisions every second (see table 7.1) [2]. This will require an even higher level of performance from the online systems than is necessary at HERA. To try to answer this challenge we have designed the first stage of a readout system capable of exceptionally high data transfer rates. To investigate

Machine	Luminosity $cm^{-2}\ s^{-1}$	Inelastic cross section	Bunch spacing	Events per second
LHC/SSC	10^{33}	100mb	25/16ns	100 million
		100nb [W,Z]		100
		0.1nb [$M_H \leq 2M_W$]		0.1
		5nb [$M_{t\bar{t}} \leq 2M_W$]		5

Table 7.1: Event rates at high luminosity colliders.

the feasibility of this system we discuss the implementation of hit finding and trigger processing within this environment.

7.1 An 80 Mbytes/s Readout System

Figure 7.1 illustrates our design, consisting of a transputer-based crate controller (CC) reading out the data from a set of readout cards (RC). This forms the basic *crate* of our system.

Each RC contains a DSP for fast data parameterisation and compaction (hit finding). This reduced data is written into a dual port memory (DPM), where

[1]This work was first presented in [1]

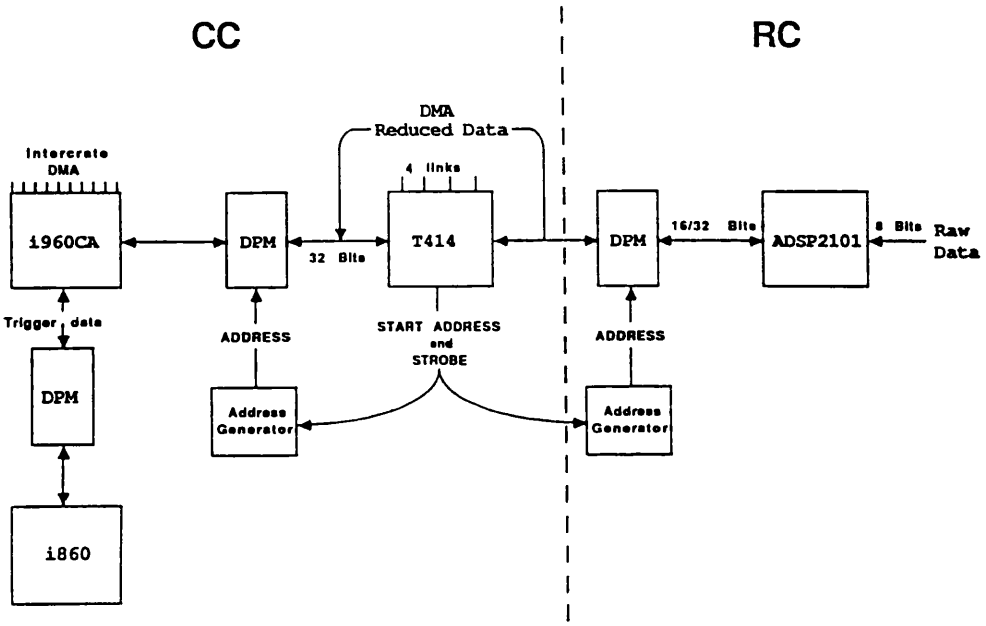

Figure 7.1: An 80 Mbytes/s Readout System.

it can be accessed concurrently by the transputer and transferred to a common DPM on the CC card. Address generators are included on the CC and each RC to enable direct memory access (DMA) operations along the crate backplane. This results in a considerable increase in the data transfer speeds obtainable, up to a projected maximum of 80 Mbytes/s.

Once the data from all RCs is assembled on the CC, it is available for trigger processing. In our system this is performed by the powerful i860 microprocessor.

The use of DPMs and a transputer on the CC as the sole controlling processor renders bus arbitration unnecessary leading to very simple interfacing logic and operating software. The four serial links of the transputer facilitate the downloading of programs and inter-crate message passing.

An important requirement for any readout system is to provide an adequate bandwidth communication link between any individual crate and the rest of the data aquisition system. To accomplish this our CC design incorporates an Intel i960CA microprocessor, whose 32-bit wide DMA channel can achieve a maximum data throughput of 106 Mbytes/s.

A full technical specification of the hardware design of our system is given in [1].

7.2 Readout and trigger processing software

The high data transfer performance of our system might cause us to question whether it can be properly matched by processing power. In this section we will try to show that it is possible to implement at least the basic software components of an online system within this high rate environment. This discussion draws on our experience within the ZEUS collaboration and addresses the problems that will be present in any similar system.

7.2.1 Hit Finding

A typical electronics crate of the type used in high energy physics has about twenty front-end cards, each supporting between 5 and 20 separate analogue readout channels. The input signals can be digitised using an FADC unit for each channel. FADCs operating at 100MHz with a precision of 8-bits currently offer the best performance in this application. Each channel is thus converted into a section of digital data typically 50 bins long. The volume of data at this stage may be considerable. Processing before readout is therefore highly desirable: rejecting unwanted noise, and parameterising pulses as hits.

This task is performed using the DSP on each front-end card. This takes the digital data as input, finds and parameterises all acceptable pulses, and writes out a small block of data associated with each hit. This might typically include

- Header information, *e.g.* channel number.
- Position of the leading edge, height, width and integrated area of the pulse and the corresponding errors.
- Special data for use by trigger processors (see section 3.2).

We shall assume here that each hit is packed into 7×32-bit words.

The peak readout speed of our system is expected to be 80 Mbytes/s. To allow the system to have some redundancy we choose an average operating rate of 40 Mbytes/s. This rate is equivalent to 1.4 Mhits/s. Assuming that there are twenty DSPs per crate, each must produce one hit every 14μs. This rate places considerable constraints on the performance of the individual DSP.

Table 5.1 in chapter 5 shows the performance we have obtained in this role with an ADSP 2100 series DSP. From these results we see that though the desired performance is very high it is just within the boundary of that which is possible.

Several other DSPs are available that might be considered for this role. For example, the AT&T DSP16A has a 33ns cycle time, while the TMS320C30 (60 ns cycle time) uses 32-bit arithmetic for all computations. We favour the ADSP 2101 because the features of its instruction set more than compensate

for its slower cycle time, while its 16-bit input registers are sufficient to deal with calculations based on the 8-bit FADC data.

7.2.2 Trigger processing

In addition to hit finding, we propose that special trigger information is also computed on each RC and transferred to the crate controller for further processing.

As an example of a realistic and non-trivial case, we discuss the use of our system for trigger processing in cylindrical tracking chambers. We assume that the chamber is placed in a constant axial magnetic field and that the primary event vertex is at the centre of the chamber.

A conformal transformation transforms circles passing through the origin of co-ordinates into straight lines in conformal space (see chapter 9). The trigger processor must solve the problem of how to recognise these straight lines.

Exploiting the processing parallelism offered by the presence of 20 DSPs per crate, we split the pattern recognition task into two steps:

1. Find short local segments of tracks.

2. Assemble these segments into full tracks.

Since each RC contains data from a small volume of the chamber, the resident DSP may perform a local search for segments. These may then be transferred to the CC where another processor may attempt to assemble them into tracks. In our design this is done by the Intel i860 microprocessor.

The simplest form of segment finding is the minimal segment finding algorithm first discussed in chapter 5 and shown to be surprisingly successful in chapter 6. Table 5.7 in chapter 5 records the performance we obtained for this algorithm. Two possible simple extensions are also illustrated in this table, an unweighted least squares fit to the segment parameters, and the calculation of the residual to the fit. As these execution times are small and are spread over the time taken to find several hits, they constitute only a small overhead to the basic DSP task of hit finding. If a more sophisticated segment finding algorithm is necessary, an extra dedicated DSP could be added on each RC.

As already stated, we propose to use the Intel i860 to find tracks from among the segments supplied by the DSPs. This microprocessor has already been discussed in chapter 4.

In conformal space the segments are described by two parameters: m and c in $Y = mX + c$. However, in the limit $P_t \rightarrow \infty$, $c \rightarrow 0$, and since one is usually interested in the tracks with large P_t, we may disregard segments whose c value is large and proceed to use the m variable for further pattern recognition. Two typical approaches are outlined below:

Clustering method: A histogram is formed based on the m value of each segment, where peaks correspond to track candidates. The histogram can either be passed on to the global trigger, or analysed at this stage. This algorithm may make use of the i860 vector library routines.

Following method: Starting from one segment, an attempt is made to follow its predicted trajectory through the chamber. New segments are added to the track candidate if they are consistent with this trajectory. After a track candidate is found, we can perform a fit to the parameters m and c, and calculate the residual to the fit before its final acceptance or rejection.

We have timed the above operations, implemented in FORTRAN, on an i860 simulator. The Intel vector library routines have been used as a convenient way to achieve high performance. Table 7.2 presents our timing results, which are based on 25 track segments per crate.

	μs
Clustering method	
Form and search histogram (5 tracks)	58.6
Following method	
Follow one track	14.9
Unweighted least squares fit (per parameter)	3.7
Calculation of fit residual (per parameter)	8.4

Table 7.2: Timings for two possible segment matching algorithms.

7.3 Conclusions

In this chapter we have shown that two basic operations are possible within our high rate environment:

- Hit finding.
- Simple trigger processing.

The feasibility of our system has been demonstrated using the minimum possible number of processors. We hope this success might encourage others to develop our system, adding to the processing power until it meets their requirements.

Bibliography

[1] R.Belusevic, G.Nixon & D.Shaw 'An 80 Mbytes/s Data Transfer and Processing System' RAL preprint RAL-90-028.
Accepted for publication in NIM.

[2] E. Eichten et al, Rev. Mod. Phys. 56 (1984) p.579.

Part III

Offline Pattern Recognition

Chapter 8

The Offline environment

Offline processing is the tool with which high energy physicists perform the final reconstruction and analysis of their data. Pattern recognition techniques find many applications in this environment. These usually demand a high level of consistency and efficiency.

This chapter is a brief review of the high energy physics offline processing environment. We shall concentrate on the challenge posed by two modern computing architectures:

- Making use of vector machines.
- Event parallelism and the use of processor farms.

We shall see that modern technology has blurred the distinction between online and offline processing.

8.1 Vector Architectures

Vector computing is of particular interest to the high energy physics community:

- It is often available on mainframe computers[1].
- It is easily accessible from FORTRAN programs.

Conventional scalar computing requires that each individual operation be completed before the next can commence. If we wish to perform the same operation for a repetitive sequence of numbers then each must wait its turn. Vector architectures divide complex operations into a series of pipelined stages. In repetitive operations, each number only has to wait for the first stage to be

[1] Vector facilities are currently available on both the CRAY X-MP and the IBM 3090 at the Rutherford Appleton laboratory.

free before beginning its journey along the pipeline. The performance of the system is now only limited by its slowest stage.

Any section of code which involves a loop and arithmetic using arrays may be suitable for vectorisation. Two conditions must be satisfied:

- The loop must access each array in regular steps.
- No calculation may refer to an array element that is altered during an earlier part of the loop.

Vectorising compilers automatically check that these conditions are satisfied before producing vectorised output. The challenge to the programmer is to design algorithms that are suitable for vectorisation.

8.2 Event Parallelism

In the previous chapters we have seen how modern computing techniques, particularly the use of highly parallel algorithms, can greatly increase computing performance. Thus, for each event the CTD SLT relies on dividing the track finding task between a large array of processors, each fulfilling a specific role. This parallel algorithm is possible because it closely follows the physical structure of the chamber and its readout.

In contrast, the offline processing environment offers different problems. The event data is no longer spread between different sub-detectors and different readout channels. The ZEUS event builder (EB) brings the data from each sub-detector into one place. This is the form in which it is eventually stored for later use. The natural parallelism that can now be exploited lies in the independence of each event. A farm of processors, each running the same sequential reconstruction program, can process many events at the same time. The total time taken for one event to pass through the system is constrained by the power of one processor, but the combined rate at which the farm completes events scales directly with the number of processor nodes it contains.

This technique can be used in any situation where the event data has already been gathered into one place and it is desired to use parallelism to improve performance. It is not necessary to perform the difficult task of designing an algorithm with internal parallelism: event parallelism can be used instead.

The ZEUS third level trigger (TLT) is based on this concept [1], and illustrates the increasing overlap between the online and offline processing environments. Positioned at the end of the online processing chain, the TLT comes directly after the EB. It will be comprised of a farm of MIPS R3000 processors. It is planned that the offline reconstruction program will be run on this farm, saving a great deal of programming effort, and providing maximum continuity between the TLT and standard offline processing.

Each R3000 processor provides an equivalent of about 10 VAX Mips. Initially a farm of about 20 nodes is envisaged, each with 16 Mbytes of memory. To allow the offline reconstruction program to run successfully in this environment it may be necessary to strip it of some of its features, reducing memory requirements and increasing processing speed.

8.3 Summary

This chapter has introduced some of the important concepts in modern offline processing. Performance benefits are possible for vectorisable algorithms, while a high degree of parallelism can be achieved by the use of processor farms. In the chapters 9 to 12 we shall follow the development of an offline 2D track finding system designed to work well in this environment.

Bibliography

[1] ZEUS collaboration 'The ZEUS detector - status report 1989', DESY March 1989.

Chapter 9

Preliminary Offline Study

[1]There are many different pattern recognition algorithms that might be used to find tracks in the ZEUS CTD [3, 4, 5, 6]. Given the detailed design of the detector and the general constraints of the offline processing environment, some of these methods will perform better than others. We shall investigate two highly contrasting strategies, both designed to find tracks in two dimensions:

- The conformal mapping of circular tracks into straight lines.
- The matching of track segments between axial superlayers.

This chapter introduces these techniques, illustrating their different approaches to the same problem.

9.1 The Conformal Mapping Strategy

A general circle in two dimensions is described by three parameters. If we constrain the circle to pass through the origin of the co-ordinate system this number is reduced to two. There are several ways of constructing a co-ordinate mapping that transforms this class of circle into a straight line in a new space.

The equation of a circle of radius R and centre of curvature (x_0,y_0) may be written

$$(x - x_0)^2 + (y - y_0)^2 = R^2 \tag{9.1}$$

We shall investigate the use of two different possible transformations.

Mapping 1

$$X_1 = x/(x^2 + y^2) \quad Y_1 = y/(x^2 + y^2) \tag{9.2}$$

[1]This work was first presented in [1, 2]

Substituting this into the equation of a circle gives

$$y_0 Y_1 = \frac{R^2 - x_0^2 - y_0^2}{2(x^2 + y^2)} - x_0 X_1 + 1/2$$

If we assume that the circle passes through the origin of co-ordinates we have $R^2 = x_0^2 + y_0^2$, and so

$$Y_1 = -\frac{x_0}{y_0} X_1 - 1/2y_0 \tag{9.3}$$

which is the equation of a straight line in X_1,Y_1 space.

Mapping 2

$$X_2 = y/x \quad Y_2 = (x^2 + y^2)/2x \tag{9.4}$$

Substituting this into the equation of a circle gives

$$y_0 Y_2 = \frac{R^2 - x_0^2 - y_0^2}{2x} + y_0 X_2 - x_0$$

Again we assume the circle passes through the origin of co-ordinates, and so

$$Y_2 = y_0 X_2 + x_0 \tag{9.5}$$

which is the equation of a straight line in X_2,Y_2 space.

In the ZEUS CTD, a typical track has an approximately circular projection on the xy plane. In addition most of these tracks will come from the small beam intersection region at the centre of the detector (see table 1.1 in chapter 1).

Taking the centre of the intersection region as our origin of co-ordinates, we can use mapping 1 or 2 to form a conformal space transforming our tracks to straight lines. We assume that a further stage of processing would then be used to find the remaining minority of tracks that are either of low momentum or do not originate from the primary vertex.

9.1.1 Basic Trials

In designing an algorithm that uses a conformal mapping technique we must address two questions:

- Which mapping should we use?
- How should we find straight lines in the conformal space?

To answer these questions PATFND was developed. Supported by its own simple CTD simulation, this prototype pattern recognition program allowed the main mapping and pattern recognition options to be tested quickly and efficiently.

Studies were carried out on a MicroVAX II. The simulation included the position resolution and two hit resolution of the CTD, but assumed a constant axial magnetic field and neglected all physics processes such as multiple scattering and energy loss. The left-right ambiguity of the CTD was only simulated in an approximate fashion. 50 event data sets were generated in the following classes:

Single tracks: $1 < P_t \leq 20$ GeV/c, $0 < \phi \leq 2\pi$, $0 < \theta \leq \pi$.

Two tracks: $P_t = 10$ GeV/c, $\theta = 2\pi$, $\Delta\phi$ =3 or 10mrads.

Multiple track events: 4-20 tracks, each with $1 < P_t \leq 20$ GeV/c, $0 < \phi \leq 2\pi$, $0 < \theta \leq \pi$.

Choice of Mapping

Mapping 1 has two distinct advantages which are not shared by mapping 2: it is manifestly unbiased with respect to the ϕ parameter of a track, and since all CTD hits have $r > 10$cm, it is always non-singular. These features led to a simpler implementation in our prototype framework, and to a more robust track finding performance. Mapping 1 is therefore chosen as our preferred conformal transformation.

Choice of Straight Line Pattern Recognition Technique

When trying to find a straight line within the conformal space of a CTD event we face a considerable combinatoric problem. To build a practical algorithm we immediately adopted three simple constraints:

- The search proceeds in an iterative manner. An attempt is made to find one track, if this succeeds the hits it contains are not considered during further processing.
- During any given iteration only hit points within a restricted ϕ region of the chamber are used. This is reasonable, as even tracks with $P_t \sim$ 300 MeV are contained in a $\Delta\phi \sim \pi/4$ region of the CTD.
- The first acceptance cut is not made on a fitted χ^2 value, but on something that may be calculated more quickly.

At any given moment, the list of hit points available to be placed on a track is called the pattern recognition *mask*. We will also make use of the concept of a *road*, the region around a track candidate where its hit points might be found. Figure 9.1 illustrates a typical CTD track before and after mapping to conformal space.

Working within our prototype framework, we have implemented and studied three different algorithms for finding straight lines in conformal space:

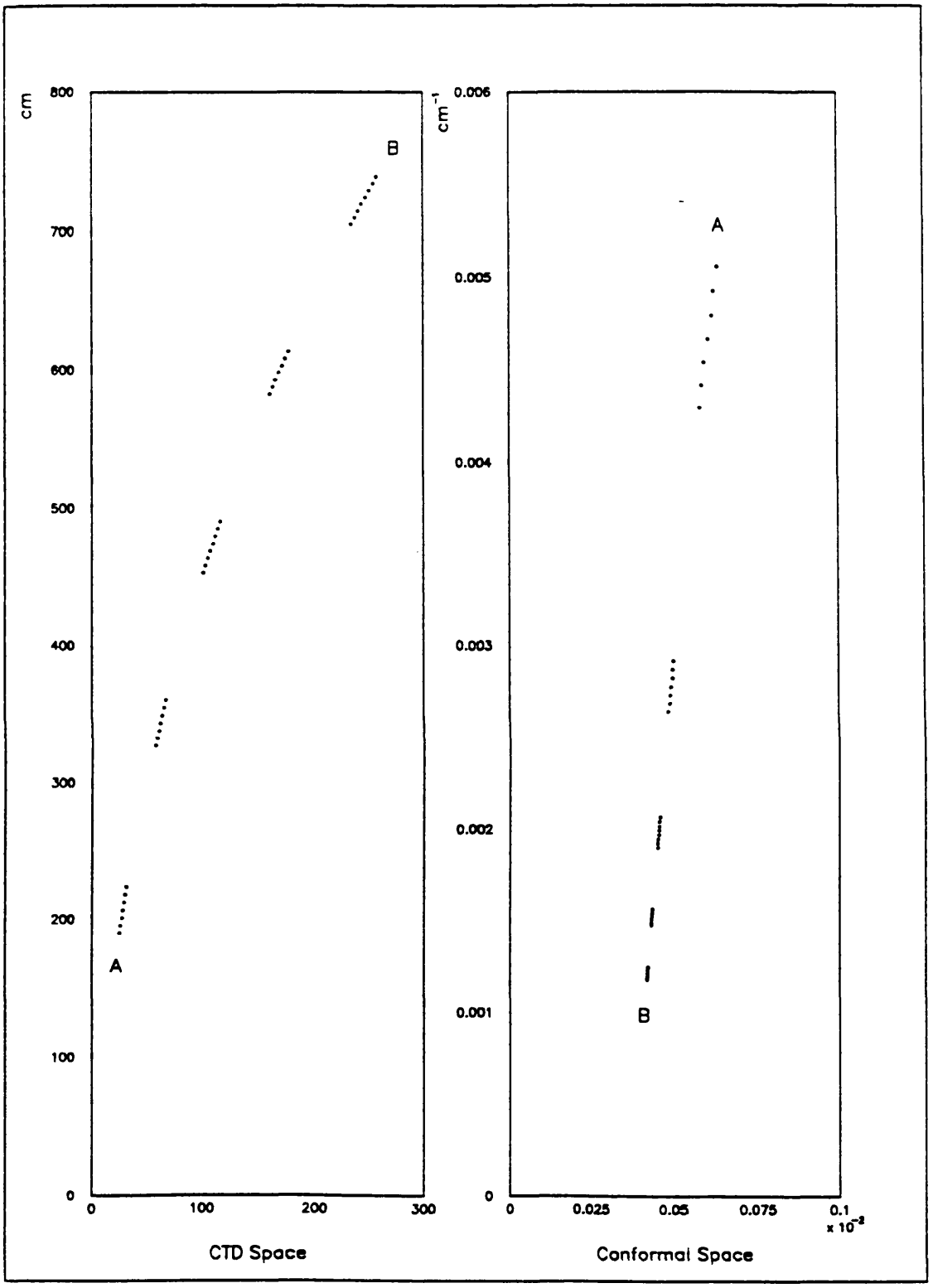

Figure 9.1: Hit points from a typical CTD track (track travels from A to B).

Table 9.1: Summary of results from preliminary conformal trials.

Method	Able to resolve close tracks ?	Time/track (ms)
1	Yes	300
2	No	20
3	Yes	60

1. From the pattern recognition mask we select the outermost hit point in real space. A road is constructed in conformal space between this and each other hit point within the mask. The road containing the largest number of hit points is used to form a track (if it passes suitable cuts).

2. We proceed as before, but now each road is formed in turn, starting with the road to the innermost hit point in real space. The first road to pass suitable cut on the number of hit points it contains is used to form a track.

3. From the pattern recognition mask we select the outermost hit point in real space. A line is constructed in conformal space between this *pivot* point and each other hit point within the mask. The angle that each of these *links* forms with the conformal X axis is entered into a histogram. The largest peak in this histogram is used to form a track (if it passes suitable cuts).

In each of these algorithms we first search the pattern recognition mask for the outermost hit point in real space. This is done to try to take advantage of the relatively low chamber occupancy in the outer superlayers of the CTD.

The results of these trials are summarised in table 9.1. Method 2 is rejected as it was unable to correctly resolve closely spaced tracks. Method 3 is preferred over method 1 because of its greater speed. This *pivot and link* method is therefore chosen as our preferred strategy.

9.1.2 Discussion of Results

The success of the pivot and link method may be understood by considering the behaviour of the link angle measurement error. Each curve in figure 9.2 is associated with a pivot point chosen at the centre of a different CTD superlayer. The variation of the link error is plotted with respect to the radius of the hit point to which this pivot is linked (all curves assume a CTD position resolution of 130μm). We observe that for a wide range of link lengths, the effective error is relatively constant. This means that when we construct the link histogram, each bin is able to act as an approximate road for a track candidate lying in the

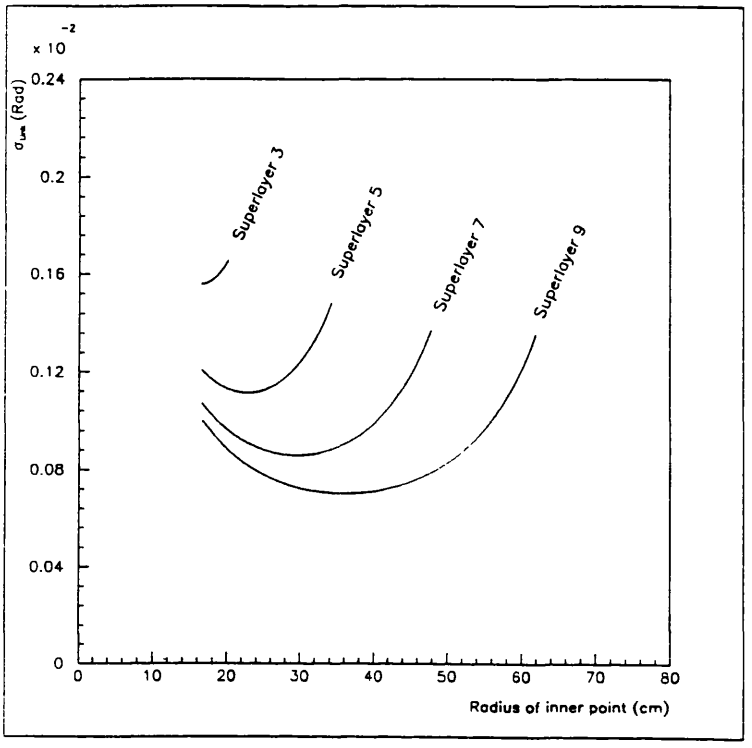

Figure 9.2: The error on a link angle as a function of the position in real space of the inner hit point: plotted for pivot points at the centre of each superlayer.

appropriate direction. For a given pivot point, this technique is a convenient way of searching all possible roads simultaneously.

Any implementation of this method involves highly repetitive loops of code: calculating the link angles, forming a histogram, searching it for peaks. The first of these tasks is generally the most time consuming. Fortunately it is also completely vectorisable, making this algorithm highly suitable for implementation on such an architecture.

9.2 The Segment Matching Strategy

An alternative approach makes use of the superlayer structure of the CTD to simplify the pattern recognition problem. Track finding proceeds in two stages:

- Find track segments within each superlayer.
- Match segments between axial superlayers to form tracks.

In chapters 4 and 5 we saw that this structure was of crucial importance to the online system, allowing a pipelined and geometrically parallel architecture. In the offline environment these considerations are not relevant, though as we shall see this approach does retain other useful features.

To complement the algorithm presented in the previous section, it is important to ensure that this alternative method does not assume that all tracks come from the primary vertex of the event.

9.2.1 Hits → Segments

Segment reconstruction has already been discussed in chapter 5 for applications within the online system. In the present application many of these concepts remain unchanged. However, for the initial tests of the offline segment matching strategy it is not important how much processing time this stage of pattern recognition requires, so long as it has a high efficiency for finding segments.

We proceed using a segment pattern recognition mask comprised of three adjacent cells. One such mask is centred on each cell within a CTD superlayer, ensuring complete coverage for almost all possible tracks. Within this mask we try to find as many segments as possible. This is achieved by making a fit to all the possible combinations of hit points that might reasonably be expected to form a segment. This procedure may be optimised to minimise execution time during a latter stage of development.

This first stage of pattern recognition greatly reduces the overall combinatoric problem which the track finder must overcome. It is also able to take direct advantage of the architectural features of the CTD described in chapter 3.

9.2.2 Segments → Tracks

As discussed in chapter 5, a track segment within a superlayer can be interpreted as a vector hit. Consider two vector hits $\mathbf{v_1}$ and $\mathbf{v_2}$, the vector joining their centres is $\mathbf{S_{12}}$. We may define the vector and dot products:

$$\begin{aligned} |\mathbf{v_1} \times \mathbf{S_{12}}| &= |\, S_{12} \,| \sin(-\alpha_1) \\ |\mathbf{v_2} \times \mathbf{S_{12}}| &= |\, S_{12} \,| \sin(\alpha_2) \\ \mathbf{v_1} \cdot \mathbf{S_{12}} &= |\, S_{12} \,| \cos(\alpha_1) \\ \mathbf{v_2} \cdot \mathbf{S_{12}} &= |\, S_{12} \,| \cos(\alpha_2) \end{aligned}$$

Following [3], we define two quantities,

$$\begin{aligned} \eta &= \frac{|\mathbf{v_1} \times \mathbf{S_{12}}| + |\mathbf{v_2} \times \mathbf{S_{12}}|}{|\, S_{12} \,|} \\ \epsilon &= \frac{\mathbf{v_1} \cdot \mathbf{S_{12}} - \mathbf{v_2} \cdot \mathbf{S_{12}}}{|\, S_{12} \,|} \end{aligned}$$

These may be simplified,

$$\begin{aligned} \eta &= \sin(\alpha_2) - \sin(\alpha_1) \\ \epsilon &= \cos(\alpha_1) - \cos(\alpha_2) \end{aligned}$$

Finally we define

$$\begin{aligned}\Gamma^2 &= \eta^2 + \epsilon^2 \\ &= 4\sin^2\left[\frac{\alpha_2 - \alpha_1}{2}\right]\end{aligned}$$

If $\mathbf{v_1}$ and $\mathbf{v_2}$ are tangents to the same circle then η and ϵ are zero. However, only their combination, Γ^2 has the property of being definitely non-zero for all $\mathbf{v_1}$ and $\mathbf{v_2}$ not fulfilling this requirement.

As already discussed, tracks trajectories inside the CTD are approximately circular when projected onto the xy plane. Γ^2 is a suitable variable to use to test whether any two track segments found in axial superlayers are consistent with coming from the same track.

9.3 Conclusions

We have discussed two contrasting pattern recognition strategies:

- Conformal mapping.
  - Assumes the position of the event vertex.
  - Suitable for implementation on vector machines.
- Segment matching.
  - Does not assume the position of the event vertex.
  - Has good combinatoric properties.

In the next chapter we shall make a full quantitative comparison of the performance of these two algorithms within the CTD.

Bibliography

[1] D.Shaw 'Conformal Mapping in the ZEUS CTD' ZEUS-UCL-88-0003.

[2] K.Long 'Some Formulae for Pattern Recognition in r, ϕ' ZEUS-RAL-88-3.

[3] W.B.Atwood. 'A Pattern Recognition Program to Study the CDC Design' SLD - New Detector Notes No.135.

[4] P.Billoir, Comput. Phys. Commun. 57 (1989) 390-394.

[5] D.G. Cassel & H. Kowalski, Nucl. Instr. and Meth., 185 (1981) 235-251.

[6] C.T.Zahn 'Graph-Theoretical Methods for Detecting and Describing Gesalt Clusters' IEEE Transactions on Comp. Vol. C-20. Jan 1971.

Chapter 10

Offline Prototype Study

[1]The 2D track finding algorithms described in the previous chapter were implemented within the ARAXNE prototype framework (see appendix C). In this chapter we shall

- Describe the implementation of each algorithm.
- Present the results of comparative trials.

10.1 Algorithms

10.1.1 Conformal Mapping Strategy

The following algorithm was implemented making use of mapping 1 from chapter 9.

1. Transform each hit point into conformal space.
2. Enter all points into a list called the *mask.*
3. Search for tracks until the mask is empty:
   (a) A broad azimuthal sector of the chamber is selected, centred on a high density of hits in the mask.
   (b) All points from the mask which are present in this sector are entered into a list called the *window*
   (c) The window is searched to find the outermost hit point in real space, which is chosen as the *pivot point.*
   (d) *Links* are constructed, in conformal space, between the pivot point and all the other hit points in the window.

[1]This study was first presented in [1]

(e) The *link histogram* is formed from the angle between each link and the conformal X axis.

(f) A *track candidate* is formed from the largest *peak* in this histogram.

(g) The track candidate is refined by adding hit points from the window lying in a road constructed between its two ends.

(h) The track candidate is further refined by dropping hit points in cases where it contains more than one from the same sense wire layer of the detector.

(i) A track candidate is *accepted* as a track if it passes cuts on the number of points it contains and the continuity of their distribution along its supposed trajectory.

(j) All points on an accepted track, along with their left-right ambiguous partners are removed from the mask.

4. *Reconstructed tracks* are formed by fitting accepted tracks to circles in the r, ϕ plane.

10.1.2 Segment Matching Strategy

Hits → Segments

A segment pattern recognition mask is defined to consist of three adjacent cells. One mask is centred on each CTD cell, ensuring that the chamber is completely covered by a series of overlapping cells.

Segment finding proceeds as follows:

1. Search each possible mask for segments.

   (a) Construct roads between all pairs of points separated by three or more wire layers.

   (b) Starting with those roads containing the greatest number of hits, fit each to $y = a + bx + cx^2$ and retain the one with the best χ^2 value.

   (c) If the absolute value of the fitted χ^2 is less than a suitable cut, we accept this candidate as a segment and delete its hits from the pattern recognition mask.

2. Remove any segments whose points are a sub-set of those found on a segment from an adjacent mask.

Segments → Tracks

The variables η and Γ^2 (see chapter 9) are used to match track segments between superlayers. Using a given tolerance for the maximum value of each of these parameters, combinations of consistent segments are assembled as track candidates. Reconstructed tracks are then formed by fitting these candidates in turn to a circle in the r, ϕ plane, and making an acceptance cut based on the χ^2 value of the fit. Once a track is accepted its segments are removed from the list of those still available.

10.2 Procedure

Events in the CTD were simulated using the SIMPLE Monte Carlo (see appendix B). This included the position resolution and two hit resolution of the CTD, but assumed a constant axial magnetic field and did not simulate physics processes such as multiple scattering and energy loss. Data sets were generated in five different classes:

Single tracks: 1000 single track events were generated in each of the following categories, P_t = 1, 4, 6, 8, 15, 30, and 50 GeV/c, $0 < \phi \leq 2\pi$, $\theta = \pi$.

Displaced vertex: 1000 single track events were generated in each of the following categories, impact parameter = 2.5, 5.0, 7.5, and 10mm In all cases the tracks had $P_t = 5$ GeV/c, $0 < \phi \leq 2\pi$, $\theta = \pi$.

Two tracks: 1000 two track events were generated with an azimuthal separation between the tracks of 15, 22, 41, 110, and 2π mrads. In all cases the tracks had $P_t = 5$ GeV/c, $0 < \phi \leq 2\pi$, $\theta = \pi$.

Neutral current events: 100 NC events were generated using LEPTO (see appendix B) at each of the following points in the HERA kinematic region,

1. $x = 0.5$, $y = 0.6$
2. $x = 0.2$, $y = 0.5$
3. $x = 0.02$, $y = 0.3$
4. $x = 0.01$, $y = 0.3$

These were chosen to vary jet energy and angle, as illustrated in figure 10.1. This allowed us to test pattern recognition performance in different characteristic regions of the HERA kinematic plane.

Data sets 1 and 2 contain events with high energy jets which only enter the first few superlayers of the CTD. These jets contain short closely spaced tracks, and are a more technically demanding test of pattern recognition performance than data sets 3 and 4. Data set 4 is characteristic of the most common type of deep inelastic events that will be observed by ZEUS.

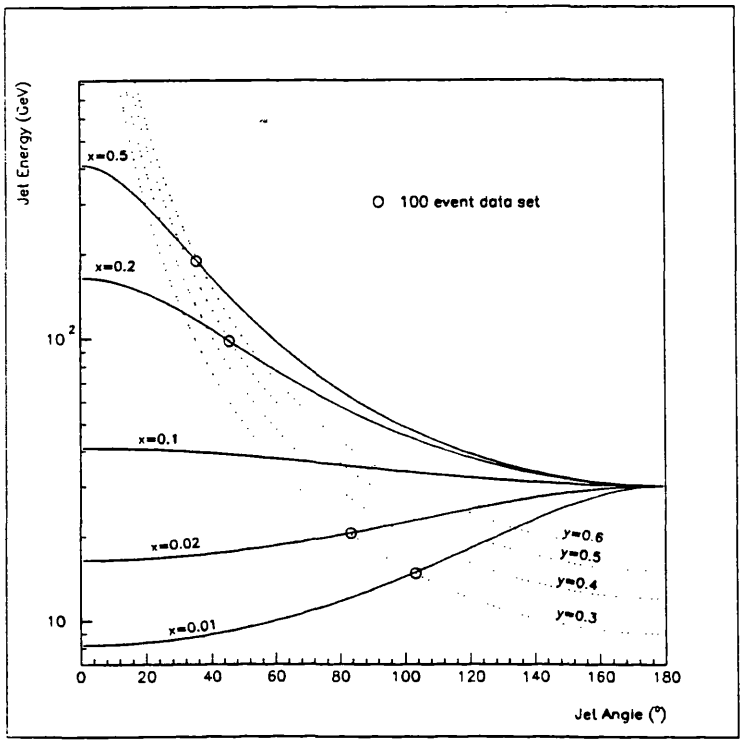

Figure 10.1: The HERA kinematic plane.

10.3 Quality and Efficiency Results

The different data sets were processed in turn by each algorithm. Our results were analysed using the techniques described in appendix A. The following fiducial cuts were used to define the Monte Carlo tracks in which we were interested:

1. $P_t > 0.4$ GeV/c
2. $19° < \theta < 161°$

The first cut ensures that we do not consider tracks which spiral inside the CTD, while the second ensures that the track reaches at least half way through superlayer 3, and so is just within the geometric acceptance of the segment matching algorithm (the conformal mapping algorithm has a slightly larger acceptance).

The results are displayed in figures 10.2 to 10.6.

- An understanding is assumed of the analysis concepts defined in appendix A.
- The global title at the top of the figure identifies the event class from which these results were produced.

- Each figure contains four graphs presenting different efficiency and quality factors as a function of the data sets within this class.
- The notation MM means that 'momentum matching' track assignment has been used to prepare this graph, while PM means that 'point matching' has been used.
- Binomial error bars are plotted for all points: if y tracks are found from a total of n, $\sigma_y = \sqrt{y(1-y/n)}$.

The individual histogram titles may be classified and explained,

Track finding efficiency: Percentage of Monte Carlo tracks that have a reconstructed track assigned to them and so are *found.*

Tracks above 96% correct: Percentage of assigned reconstructed tracks that have more than 96% of the hits from their associated Monte Carlo track placed on them (for a CTD track at $\theta = 90°$, 1 hit $\sim 2.5\%$).

Tracks contaminated by ghosts: Percentage of assigned reconstructed tracks for which at least one point from the Monte Carlo track has been placed on them with the incorrect sense of the left-right ambiguity.

Tracks contaminated by other points: Percentage of assigned reconstructed tracks that are contaminated with a least one point from a different Monte Carlo track.

Spurious tracks: Reconstructed tracks not assigned to any Monte Carlo track.

Tracks split: Generated tracks that have been split into two or more reconstructed tracks.

Scattered electron efficiency: In NC events, the efficiency for finding the current electron.

The reader may be guided through the interpretation of these figures:

Single track events (Figure 10.2) Both algorithms achieve a similar track finding efficiency. However, we observe a tendency for the segment matching algorithm to find a higher percentage of its tracks with more than 96% of their correct hits, while the conformal mapping algorithm yields a lower rate of contaminated tracks. This is explained by a differing strategy towards the acceptance of ambiguous/borderline points: if two points from a single wire layer were consistent with a given track, then the segment matching algorithm has been implemented to accept the better point, while conformal mapping algorithm rejects both. This is a trend repeated throughout these results.

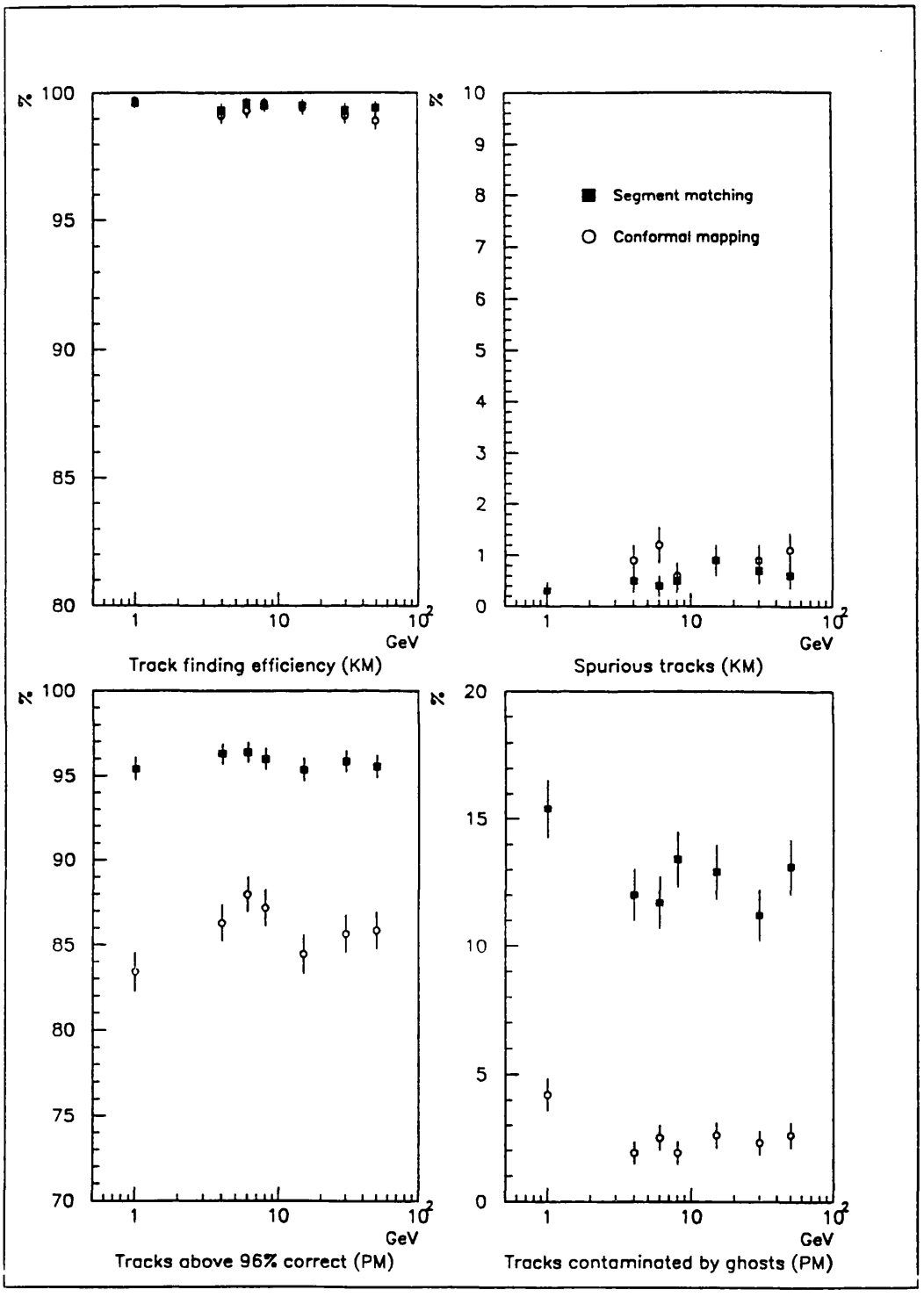

Figure 10.2: Single tracks of fixed P_t.

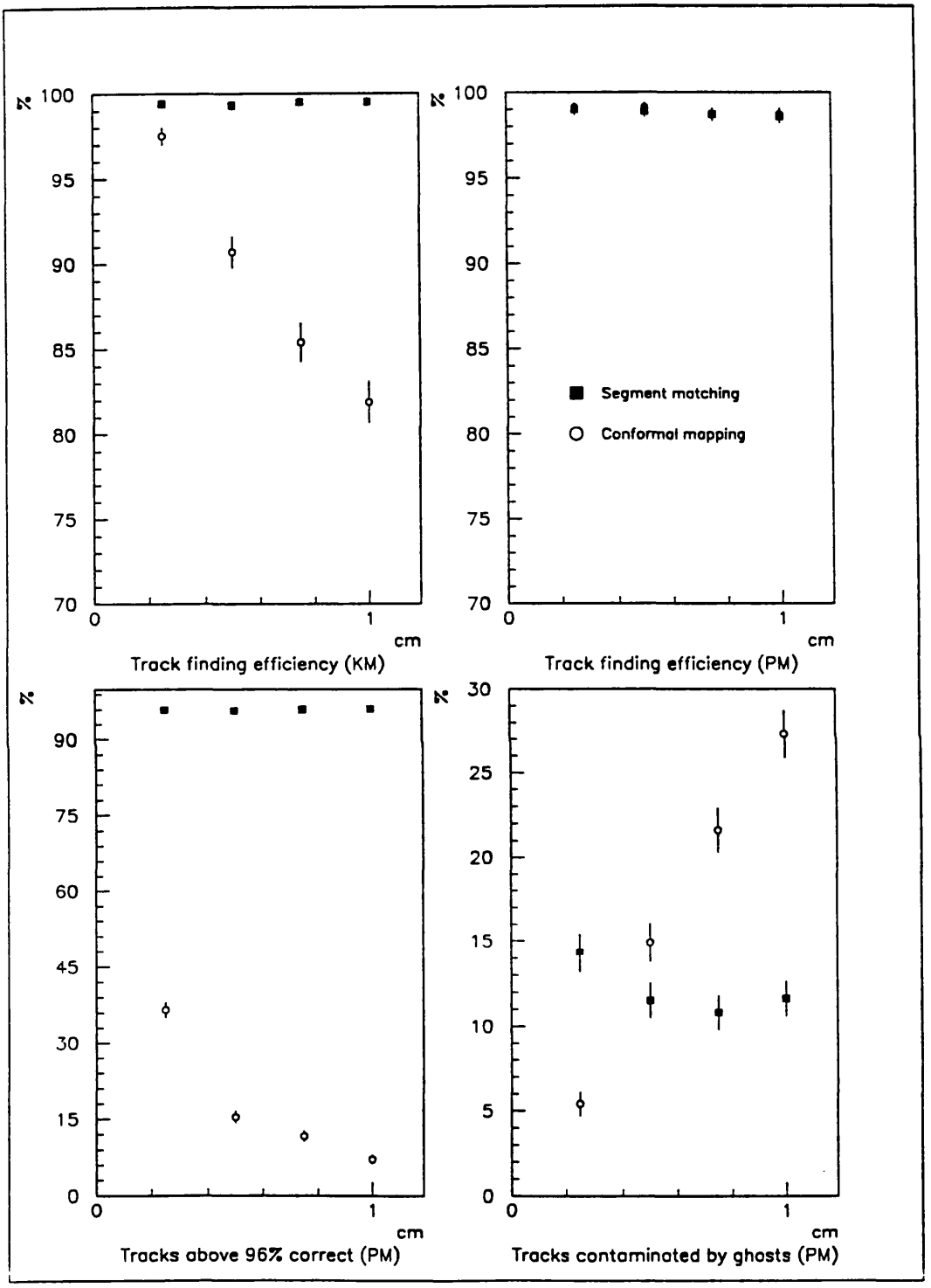

Figure 10.3: Single tracks with a displaced vertex.

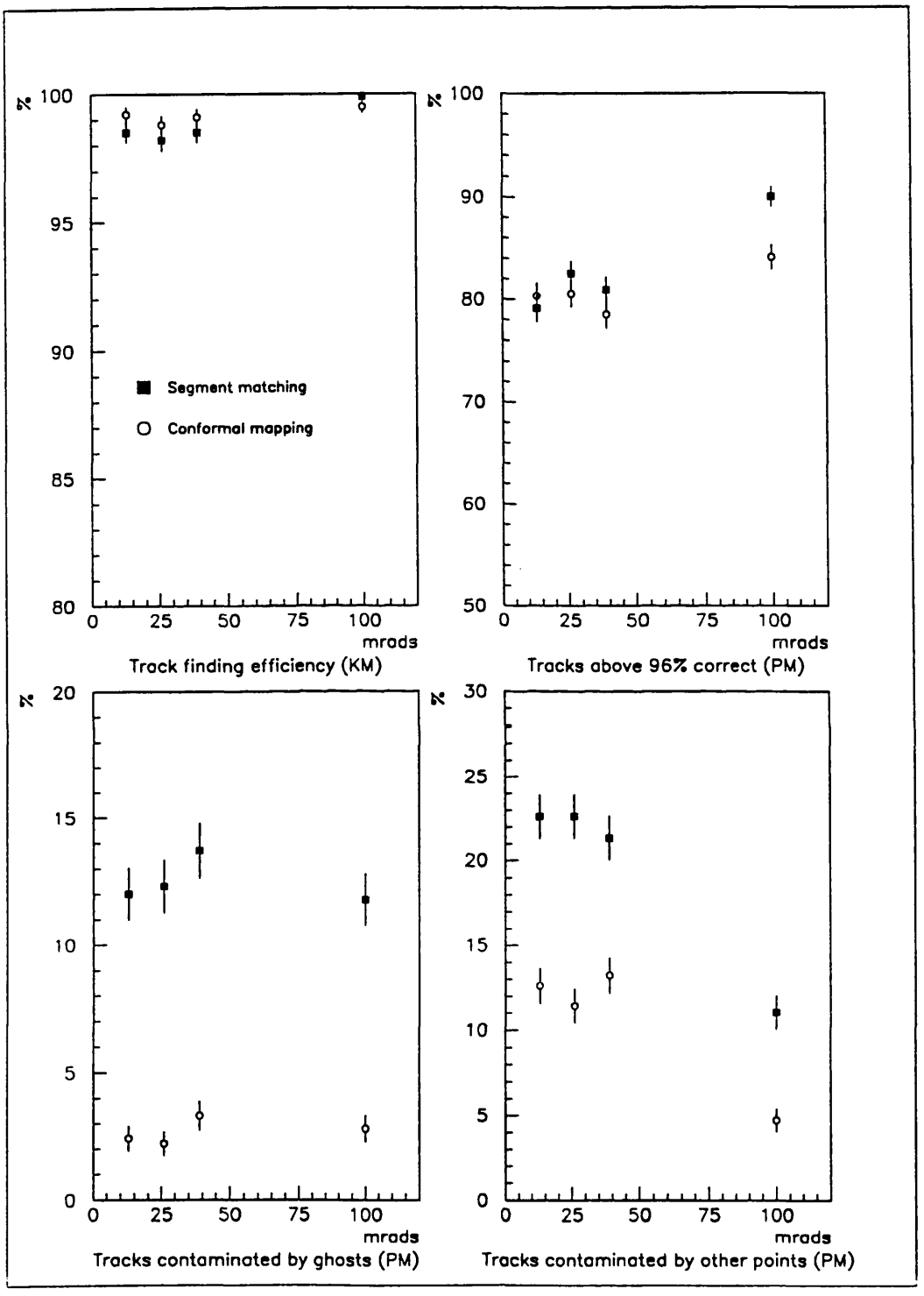

Figure 10.4: Two tracks separated by a small angle.

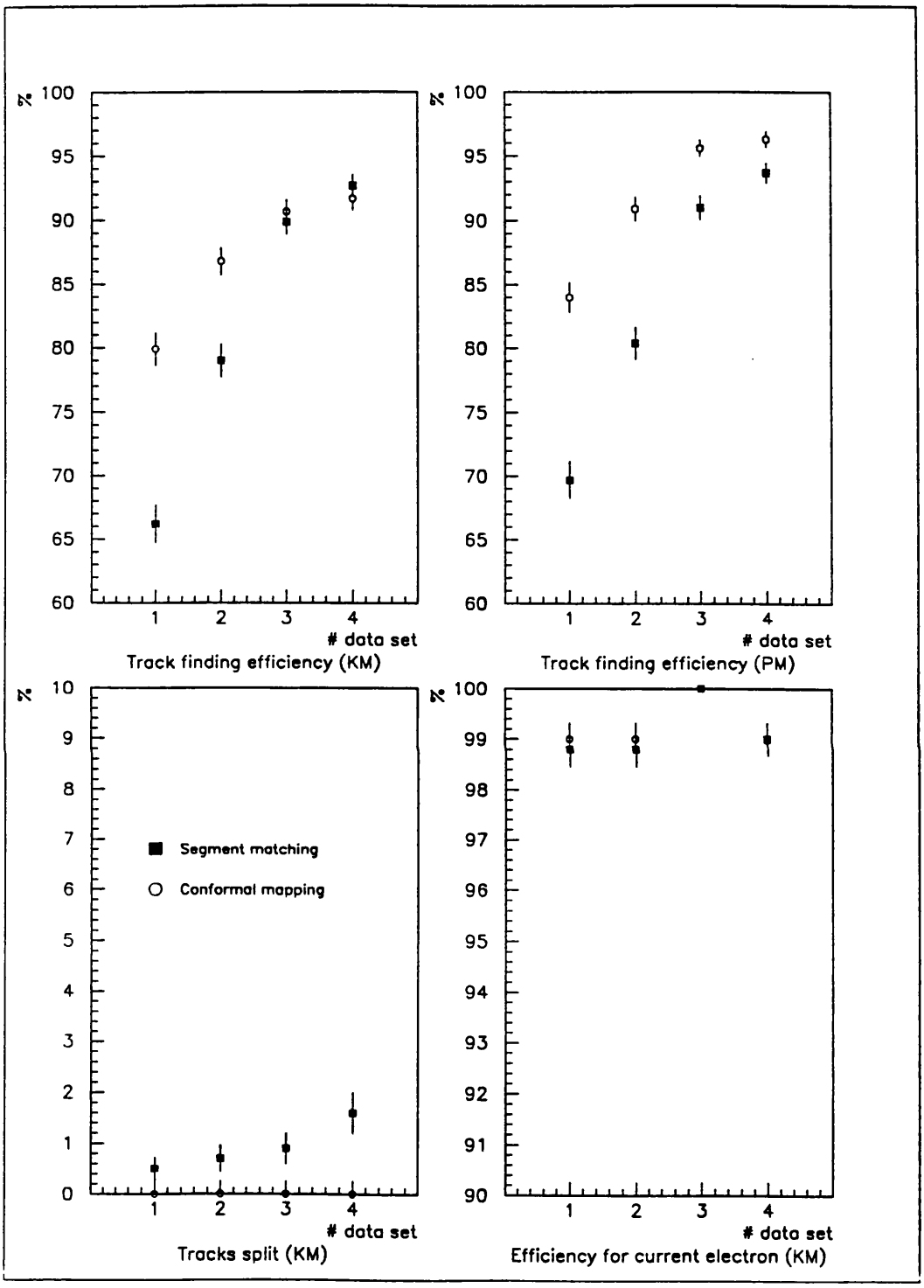

Figure 10.5: Neutral current events at characteristic points in the HERA kinematic plane: efficiency factors.

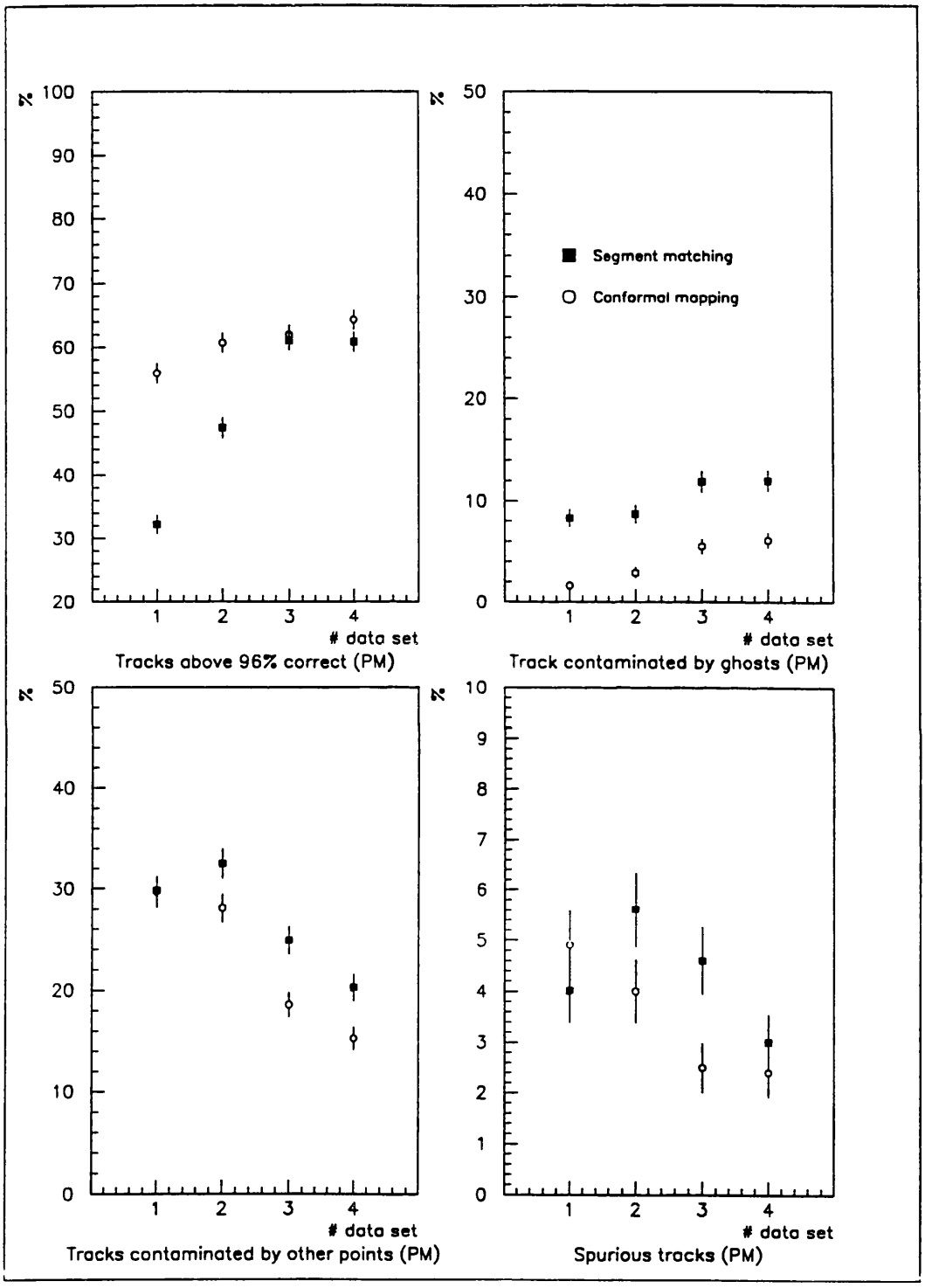

Figure 10.6: Neutral current events at characteristic points in the HERA kinematic plane: quality factors.

Displaced vertex events (Figure 10.3) The conformal mapping technique assumes knowledge of the event vertex. We therefore expect its efficiency to fall off as a function of the displacement of the true event vertex from its nominal value. This is observed in these histograms and represents the major drawback of this method.

Two track events (Figure 10.4) Both algorithms perform similarly when presented with the problem of reconstructing two closely spaced tracks. Each suffers a clear deterioration in performance even for separations of up to 100mrads.

Neutral current events (Figures 10.5 and 10.6) These figures show the results for the neutral current data sets. While both algorithms show a deterioration in performance while handling tracks within jets, it is clear that conformal mapping does systematically better in the data sets at higher jet energies. It is encouraging to note that both algorithms retain a very high efficiency for identifying the current electron.

10.4 Discussion of Results

Our results can be summarised by two main points:

- The conformal mapping results clearly demonstrate the disadvantage of assuming that all tracks come from the nominal interaction region. When the position of the real vertex is displaced, this method increasingly finds tracks with biased kinematic parameters and of lower general quality.
- The efficiency and quality of the segment matching approach falls more rapidly than that of conformal mapping approach in the more difficult part of the NC kinematic plane. It is desirable to minimise this systematic reduction in performance and eliminate any effects it might have on physics analysis.

Unfortunately these points lead us in conflicting directions: while not wanting to include the event vertex bias of the conformal mapping algorithm in our reconstruction program we are unable to choose the segment matching approach due to its unacceptably poor performance in some regions of the HERA kinematic plane.

10.5 Conclusions

When this study was originally conceived it was hoped that it would allow us to adopt one of the algorithms under trial for implementation within the final

ZEUS reconstruction program: this has not been possible.

We conclude that a new design is necessary, combining the best features in each of our prototype algorithms and drawing on the experience gained in these trials.

Bibliography

[1] D.Shaw, K.Long, & S.Tkaczyk, 'A Comparative Study of Two Prototype Pattern Recognition Algorithms for the ZEUS CTD' ZEUS-UCL-89-0004.

Chapter 11

The Offline Design

[1]This chapter aims to provide an insight into all stages of the final CTD track reconstruction system. It includes

- An overview of the complete CTD track reconstruction methodology.
- A detailed presentation of the CTD 2D track finding system.

This design is a product of the experience gained from the work presented in the previous chapters of this thesis.

11.1 Track Reconstruction Methodology

11.1.1 Two Dimensional Track Finding

Track finding proceeds in two stages: First a track seed is found, specifying the trajectory of a candidate track at a given point in the chamber, this is then extended backwards using a progressive track following algorithm.

Seed Finding

A seed is defined to have the same kinematic variables as a track, expressed at a convenient reference surface near to where the track is thought to leave the chamber (see chapter 3).

The seed finder is restricted to search only the list of hit points defined by the *seed mask*. All hits that have already been placed on a track candidate are removed from this mask.

A principle feature of our design is that any number of different algorithms may be employed to generate seeds. The choice of algorithm may be steered during the course of processing an event, ensuring maximum robustness and flexibility.

[1]This design was first presented in [1]

Conformal seed finding Based on the conformal transformation discussed in the two previous chapters, this seed finder aims to make global use of the hits in the chamber. Though this method is highly suitable for vectorisation, it has the disadvantage of assuming that all tracks come from the nominal interaction point, and so it is not kinematically unbiased.

Segment seed finding This seed finder works by reconstructing a track segment from among the hit points in an outer superlayer. Fitting these points to a circle yields the kinematic parameters of a seed. The pattern recognition involved in this type of algorithm is inherently very fast because of the low combinatoric background. This method is kinematically unbiased, and can even be used for tracks of very low momentum.

Seed Extending

Taking the seed as a starting point, a track candidate is formed by extrapolating back along the trajectory it defines, stepping inwards to each axial superlayer in turn. After each step, any hit points falling within a locally defined road are accepted as belonging to the candidate. The track parameters are continually updated using a Kalman filtering technique [2].

During the seed extrapolation the track candidate is free to use any hit point falling within the road it defines, regardless of whether that point has already been placed on another track candidate. The progressive nature of the track search is ensured because all hits used by a new track candidate are deleted from the seed mask (see previous section).

11.1.2 Three Dimensional Track Finding

The even numbered superlayers contain stereo wires. The azimuthal displacement between a segment found in one of these layers and the track to which it belongs is a measure of its z co-ordinate. Trying to match such segments to two dimensional tracks allows track reconstruction to be completed in three dimensions.

11.1.3 Track Parameters Estimation

Once we have found all possible track candidates in three dimensions, we resolve any ambiguities arising where two tracks share a common point. We can now apply detailed corrections to individual hit point measurements based on their position in z and the trajectory of their parent track.

The final estimate of the track parameters is made using the Kalman Filter formalism. This was first published in [3] and discussed for track fitting applications in [4].

The method has several advantages.

1. It needs only to invert matrices of the order of the individual measurement vectors.

2. The progressive nature of the technique allows us to recognise kinks in tracks and to reject outlying points even at this final stage.

11.2 CTD 2D track finding system

CTD 2D track finding is illustrated in detail in figures 11.1 to 11.6. This is just one part of the complete system described in the previous section. These figures have been prepared using SASD techniques (appendix D). Processes are represented as 'bubbles', each of which may in turn be expanded to reveal another set of bubbles representing its sub-processes. The figures in this section represent several layers of such a structure.

(1) Find tracks in two dimensions Hits in the axial superlayers are searched for 2D track candidates.

(1.1) Find seed Hit points that have not yet been used or rejected by the track finding procedure are present in the seed mask. During each iteration of the track finder, the current seed mask is searched to try to find one or more seeds. Regardless of whether this search is successful at least one point must be rejected by being deleted from the seed mask (this precaution ensures that the same current mask can never be passed to a seed finder twice, and so an infinite loop cannot occur at this point).

(1.1.1) The Gardener Given the current status of the 2D track search, the gardener decides which algorithm shall be used to search the current seed mask. The flexibility and robustness that this choice brings to the track finder is a feature of this method.

(1.1.2) Find conformal seed This seed finding algorithm is based on the conformal mapping technique developed in chapters 9 and 10. This procedure always deletes one or more hits from the seed mask, and may produce a seed.

(1.1.2.1) Find pivot hit The hit furthest from the centre of the CTD is chosen as the pivot hit. The left-right ambiguous points associated with this hit are the pivot *points*, which are immediately deleted from the seed mask.

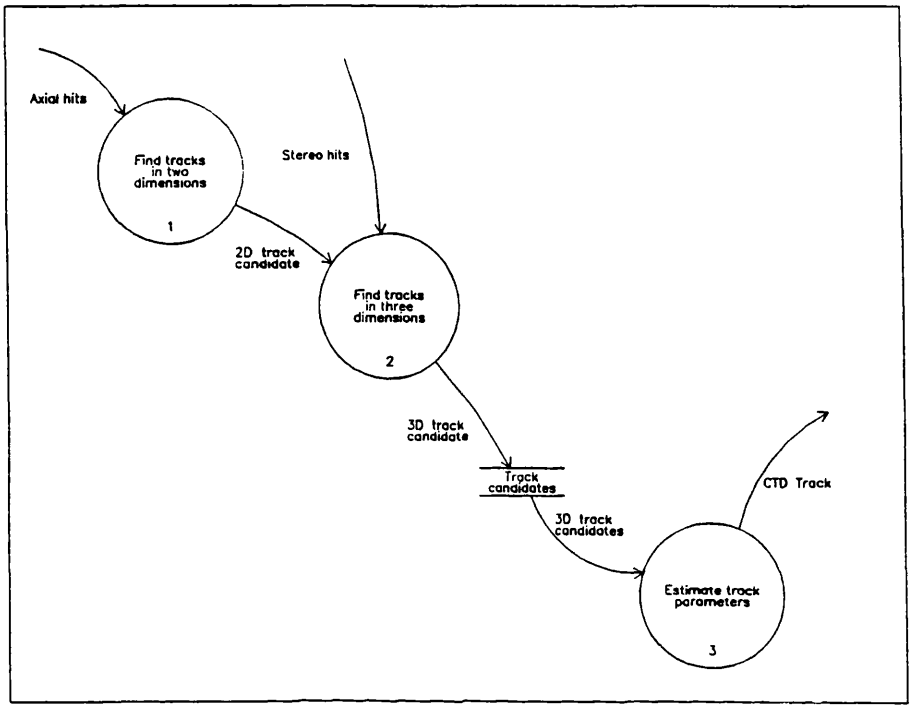

Figure 11.1: CTD track reconstruction.

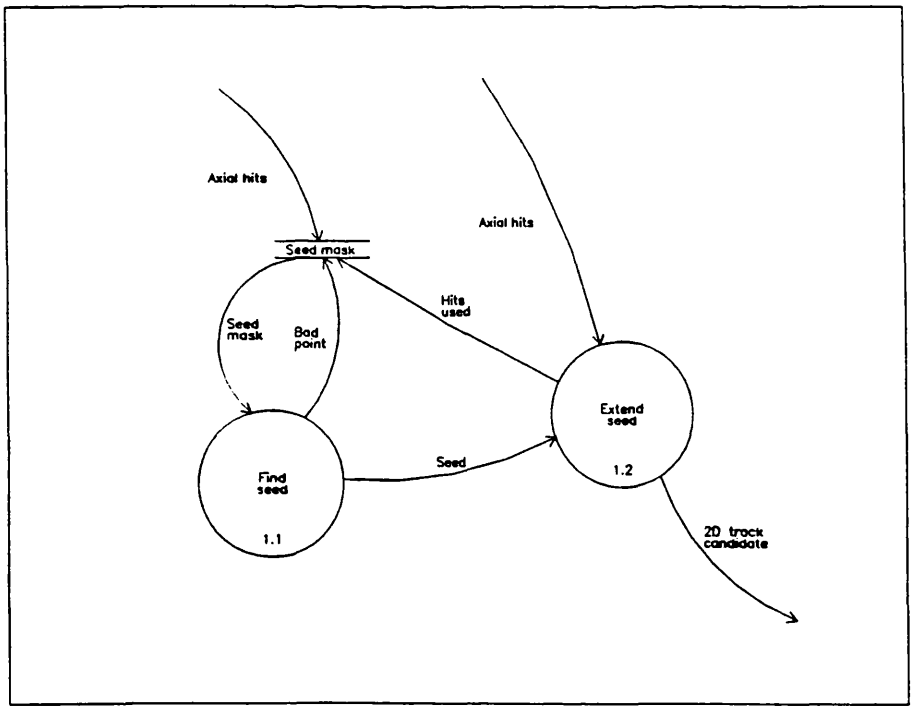

Figure 11.2: Find tracks in two dimensions.

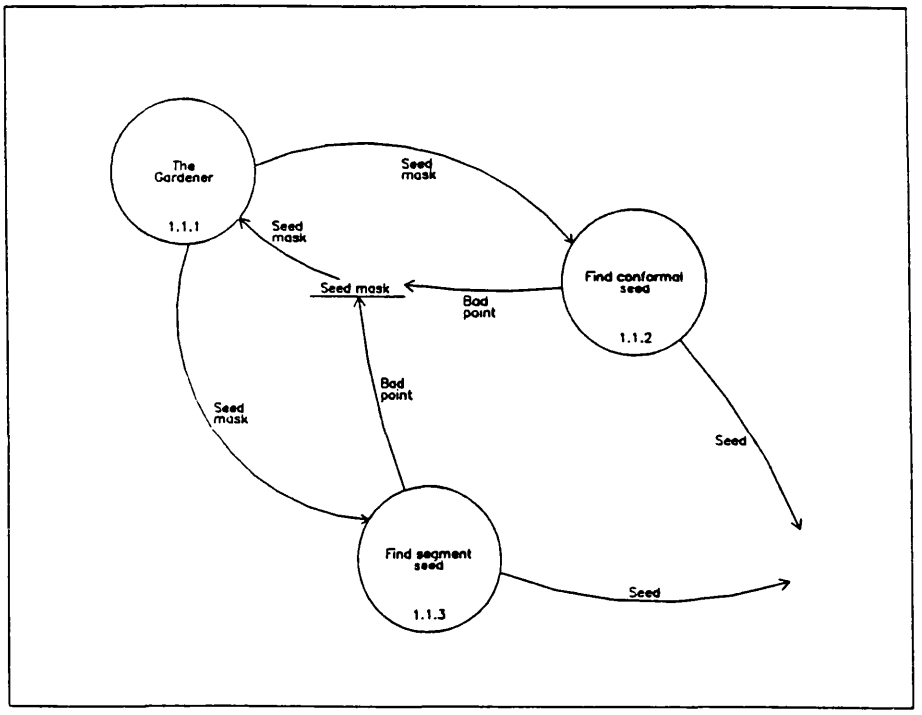

Figure 11.3: Find seed.

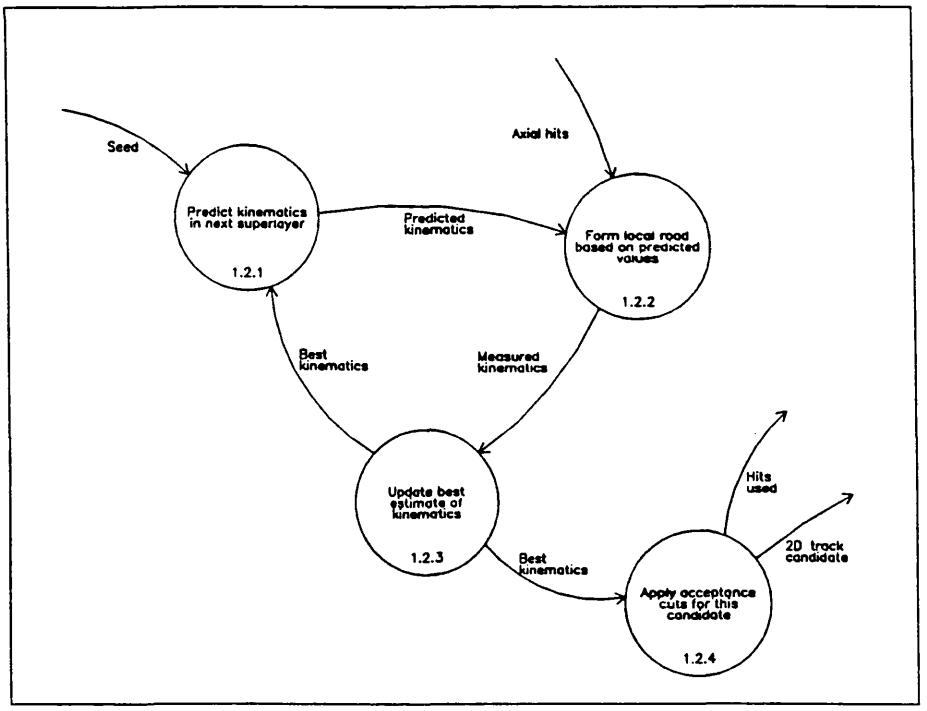

Figure 11.4: Extend seed.

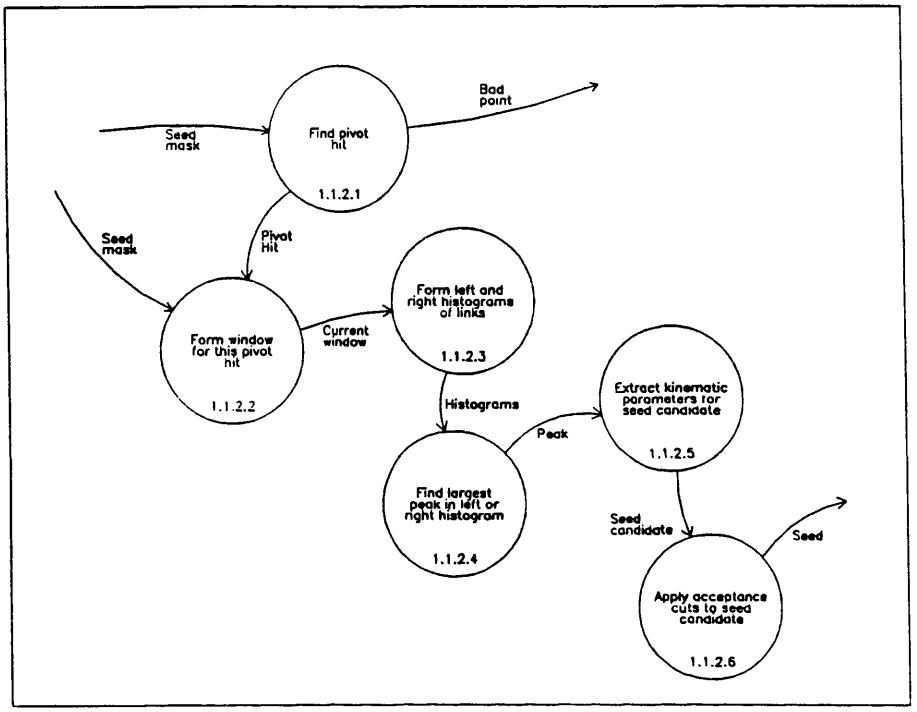

Figure 11.5: Find conformal seed.

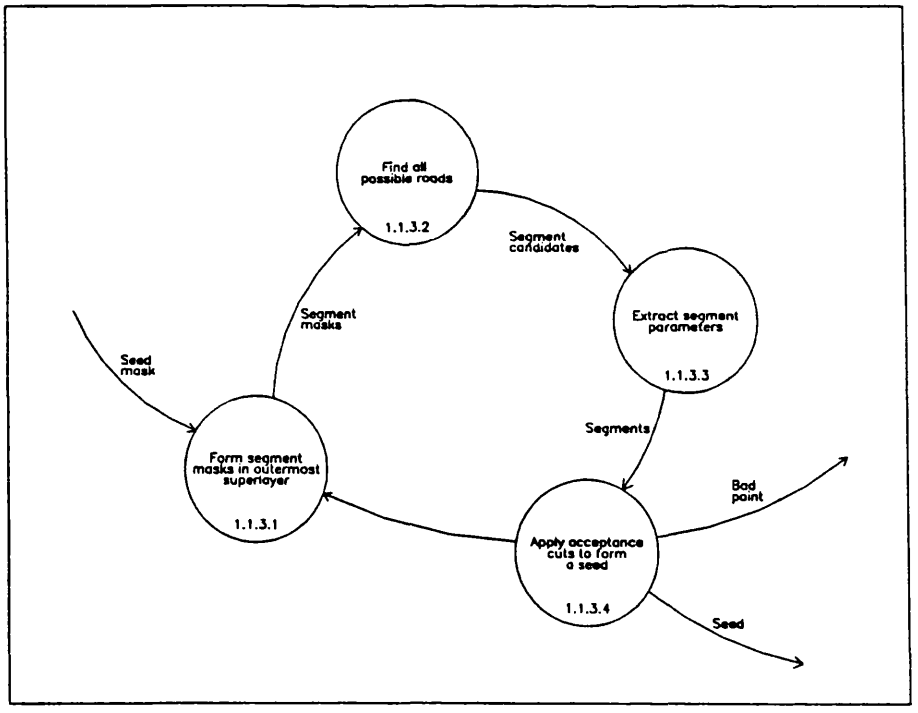

Figure 11.6: Find segment seed.

(1.1.2.2) Form window for this pivot hit An azimuthal sector of the CTD within $\pm\pi/2$ of either pivot point is defined. This region is bounded by the innermost wire layer and extends up to, but does not include, the wire layer of the pivot points. Any points in the seed mask which are also in this area are placed in the current *window*. Only points from within the current window will be used during this seed finding attempt.

(1.1.2.3) Form left and right histograms of links Taking each pivot point in turn, a link is constructed in conformal space between it and each point in the window. One histogram is formed for the angle of links to the left pivot point, another for all links to the right pivot point.

(1.1.2.4) Find largest peak in left or right histogram The largest single bin in either histogram is found. The histogram *peak* is defined to comprise of this bin plus a given number of bins on each side.

(1.1.2.5) Extract kinematic parameters for seed candidate The parameters that would be needed to specify a seed are found. These may be obtained directly from the position of the peak, or by fitting the points it contains to a circle in real space.

(1.1.2.6) Apply acceptance cuts to seed candidate The seed candidate is only accepted as a seed if it passes a cut on the number of hits within its peak as a fraction of the pivot hit wire layer. If a circle fit has been used to derive the seed kinematic parameters, a cut is also made on the fitted χ^2 value.

(1.1.3) Find segment seed This seed finding algorithm finds seeds based on track segments found within a superlayer. This always deletes one or more hits from the seed mask, and may produce one or more seed candidates.

(1.1.3.1) Form segment mask A *segment mask* is defined to contain a centre cell plus a given number of cells on either side. The outermost superlayer that contains any hits in the current seed mask is found, and all possible segment masks in this superlayer are formed.

(1.1.3.2) Find all possible roads For each segment mask in turn a straight road of given width is formed between all the eligible pairs of points it contains. To be eligible a pair of points must be at least three wire layers apart. Never allowing more than one hit per wire layer, all combinations of hits falling within a road are noted as segment candidates.

(1.1.3.3) Extract segment parameters A circle fit is made to each segment candidate. This estimates the segment kinematic parameters and provides a χ^2 value which measures the probability of the fit to these points.

(1.1.3.4) Apply acceptance cuts to form a seed Once all possible segment masks in the outermost superlayer have been processed, the segment containing the most hits and also passing a cut on its χ^2 value is accepted as a seed.

(1.2) Extend seed Taking the seed as its starting point, an attempt is made to form a 2D track candidate using all the axial hit points in the chamber. If the attempt is successful, the track candidate is stored and all its hit points (and their left-right ambiguous partners) are deleted from the seed mask. This ensures that the track we have found will not give rise to any more seeds.

(1.2.1) Predict kinematics in next superlayer The intersection of the current track candidate with the next superlayer is predicted (if this is the first prediction for this candidate then the seed kinematics are used as input).

(1.2.2) Form local road based on predicted values Within the new superlayer a *road* is formed around the predicted trajectory of the track candidate. The width of this road is determined by the position resolution of the chamber and the calculated error on the extrapolated kinematic parameters. For each wire layer up to one hit within the road may be placed on the track candidate. If there are two hits from the same wire layer within the road then only the one closest to its centre is taken.

(1.2.3) Update best estimate of kinematics The kinematic parameters of the track candidate are estimated using all the hits that have so far been placed on it. If estimated errors on these new kinematic values mean that they are no longer consistent with the predicted values for this superlayer, then they are recorded as the present *best* estimate of the kinematic parameters, otherwise the predicted values are recorded as the best estimate.

(1.2.4) Apply acceptance cuts for this candidate After the innermost superlayer has been searched for hits, a 2D track candidate is only recorded if it passes certain acceptance cuts. These are made on the number of hits it contains as a fraction of the wire layer of its outermost hit, and on the χ^2 value of the fit to its final best kinematics parameters. Any hits used on a successful candidate are deleted from the seed mask.

	μm
Conformal Seeds:	
ϕ	37
ψ	56
Q/P_t	14
Segment Seeds:	
ϕ	55
ψ	490
Q/P_t	5300

Table 11.1: Effect of seed parameter errors on road width after extrapolation to the next superlayer.

11.2.1 Comparison of seed finding techniques

The best way to compare the two alternative seed finding techniques is to investigate the absolute accuracy with which they measure the seed kinematic parameters. It is these parameters that usually form the basis for the first inward extrapolation step from the superlayer in which the seed is found. Table 11.1 presents estimates for the contribution the measurement error on each kinematic parameter makes to the error on a drift distance predicted in the new superlayer. These estimates have been made using tracks simulated in the CTD with $P_t = 10$ GeV/c and $\theta = \pi/2$, allowing the conformal seeds to be based on a fit to around 40 points, while the segment seeds are based on only 8 points. All the values quoted remain roughly constant with changes in momentum.

It would be desirable for none of these errors to exceed the position resolution of the chamber. This is not the case for segment seeds. The error on Q/P_t for segment seeds is large because this type of seed has very little information with which to measure track curvature. We can decrease the value of this error by constraining the track to come from the nominal interaction region, or simply by assuming that the track is of infinite momentum. However, both of these techniques introduce a kinematic bias and so are undesirable, and in any case the error on ψ cannot be reduced in this fashion.

11.2.2 Discusion of Design

The decomposition of 2D track finding into two parts is designed to allow the greatest possible pattern recognition flexibility: new algorithms can be conceived at any time for use as seed finders or seed extenders. Allowing the 'Gardener' to dynamically steer the choice of seed finding algorithm further extends this inherent flexibility.

To face the combinatoric problems associated with CTD track finding we will often want to use 'fast' seed finding algorithms. To achieve their high speed these will often make assumptions that tend to bias the kinematic parameters of the seeds which they find. It is the role of the extend seed stage to filter out such kinematic imperfections, and to ensure that only track candidates of a certain minimum quality are accepted. The seed extender is able to do this because it is isolated from the earlier combinatoric difficulties, and so can be implemented in a kinematically unbiased way, while retaining an acceptable execution speed.

As we saw in chapter 8, it is planned to use the offline reconstruction program to implement the ZEUS TLT. This may require that a special 'fast' version of the program is available. Our new design is ideally suited to this role, allowing us to make use of many different seed finders, including ones specially adapted for the TLT. One option would be to derive seeds from SLT tracks, allowing an immediate partial reconstruction of the event. In all cases we could rely on our extend seed algorithm to ensure that our tracks were of a consistently good quality.

11.3 Conclusions

In this chapter we have presented a new design for the CTD tracking system, with particular emphasis on 2D track finding. Building on our previous experience, this combination of algorithms allows great flexibility and is suitable for adaptation to the TLT environment.

Bibliography

[1] N.Dyce et al, 'Track reconstruction methodology in the ZEUS Central Tracking Detector' Conf. Computing in HEP, Santa Fe, 1990.

[2] P.Billoir, Comput. Phys. Commun. 57 (1989) 390-394.

[3] R.E.Kalman, J. Basic Eng. 82 (1961) 34.
R.E.Kalman and R.S.Bucy, J. Basic Eng. 83 (1961) 95.

[4] R.Frühwirth, Nucl. Instrum. Methods 262 (1987) 444.

Chapter 12

Offline Performance Study

[1]The CTD 2D track finding system described in the previous chapter has been implemented within ZEPHYR, the ZEUS physics reconstruction program (see appendix D). In this chapter we shall:

- Study the performance of this new track finding system.
- Present execution times for the vectorised conformal seed finder.

12.1 Algorithms

The algorithms described in chapter 11 have been implemented within ZEPHYR. In this preliminary version the following features were incomplete:

- Only conformal seed finding was available.
- All fitting was done using an iterative circle fitter [2], no Kalman techniques were used.
- During extend seed steps, the errors on extrapolated track parameters were neglected.

12.2 Procedure

Detector response was simulated using MOZART (see appendix B). This included the position resolution and two hit resolution of the CTD, but assumed a constant axial magnetic field and neglected all physics processes such as multiple scattering and energy loss. This was done for two reasons:

- To allow a direct comparison with the results presented in chapter 10. These used the SIMPLE Monte Carlo and did not simulate these features.

[1]This study was first presented in [1]

- Tracks and hits generated in this way are statistically well behaved. This greatly facilitates program testing (see appendix D).

Data sets were generated in three different classes:

Single tracks: 1000 single track events were generated in each of the following categories, $P_t = 1$, 3, 10, 32, and 100 GeV/c, $0 < \phi \leq 2\pi$, $\theta = \pi$.

Two tracks: 1000 two track events were generated with an azimuthal separation between the tracks of 10, 50, 100, and 2π mrads. In all cases the tracks had $P_t = 10$ GeV/c, $0 < \phi \leq 2\pi$, $\theta = \pi$.

Neutral current events: 100 NC events were generated using LEPTO (see appendix B) at each of the following points in the HERA kinematic region,

1. $x = 0.5$, $y = 0.6$
2. $x = 0.2$, $y = 0.5$
3. $x = 0.02$, $y = 0.3$
4. $x = 0.01$, $y = 0.3$

These data sets have been chosen to allow direct comparison with those used as part of the study in chapter 10. As before, they allow us to test pattern recognition performance across a wide region of the HERA kinematic plane.

In addition, options were implemented within ZEPHYR that allowed us to perform other tests with this data:

- The conformal seeder assumes a knowledge of the position of the nominal interaction region. By deliberately moving the assumed point we can simulate tracks not coming from the primary vertex.
- By selectively dropping hits from particular superlayers in the chamber, we are able to simulate tracks at different values of θ.

12.3 Quality and Efficiency Results

The different data sets were processed in turn by our system. The results were analysed using the techniques described in appendix A. The following fiducial cuts were used to define the Monte Carlo tracks in which we were interested:

1. $P_t > 0.45$ GeV/c
2. $19° < \theta < 161°$

These cuts are consistent with those chosen in chapter 10.

Figures 12.1 to 12.6 are designed for direct comparison with the corresponding figures from chapter 10, where their presentation and titles are explained. The reader may be guided through the interpretation of these figures:

Single tracks (Figure 12.1) The performance of the track finder is very good. The only notable defect is a slight depression for the $P_t = 1$ GeV/c events in the 'Tracks above 96% correct' graph. This is thought to be caused by our present failure to extrapolate errors estimates during the individual extend seed steps.

Inclined tracks (Figure 12.2) This is an extra study, not present in chapter 10. The first data point represents tracks only just reaching superlayer 3 in the CTD, while the last data point represents tracks just reaching superlayer 9. The fall in the track finding efficiency for the shortest tracks is clearly due to the rising contamination from ghost points and the resultant rise in spurious tracks.

Displaced vertex (Figure 12.3) Even though conformal seeds have been used, the sensitivity of the track finding performance to a displacement of the event vertex is greatly reduced compared to that in chapter 10. Note the different distance scale used in this chapter.

Two tracks (Figure 12.4) Again the performance is encouraging. The first data point corresponds to the closest tracks, which are separated by only 10mrads (this is closer than was attempted in chapter 10). At this distance the tracks overlap significantly in the first superlayer. This is reflected in the high track finding efficiency achieved despite the contamination on 20% of tracks.

Neutral current events (Figures 12.5 and 12.6) The increase in the track finding performance for all our NC data sets is very encouraging. It is pleasing to note that the kinematic matching and point matching analyses are now in exact agreement. The increased quality of these tracks compared to chapter 10 is marked. The percentage of tracks found with more than 96% of their true points is depressed by two factors: the general complexity of these events leading to pattern recognition ambiguities, and the effect observed in figure 12.1 for $P_t = 1$ GeV/c tracks.

These results are a clear improvement on those presented in chapter 10, and are a vindication of our design process.

Though the results are good, we would still like to understand the loss in track finding efficiency we observe in NC event sets 1 and 2. As can be seen from figure 10.1 in chapter 10, these data sets contain high energy jets with

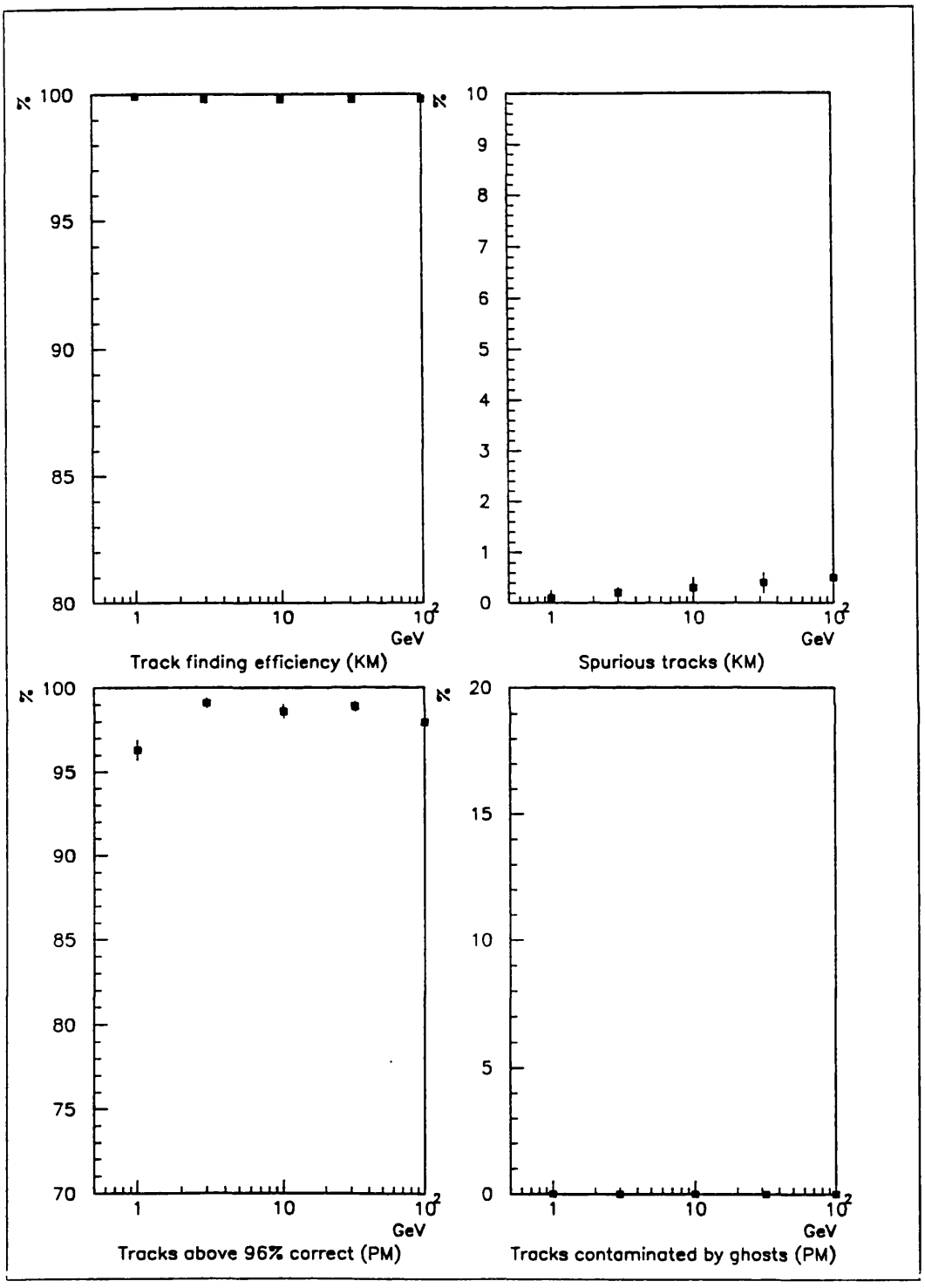

Figure 12.1: Single tracks of fixed P_t.

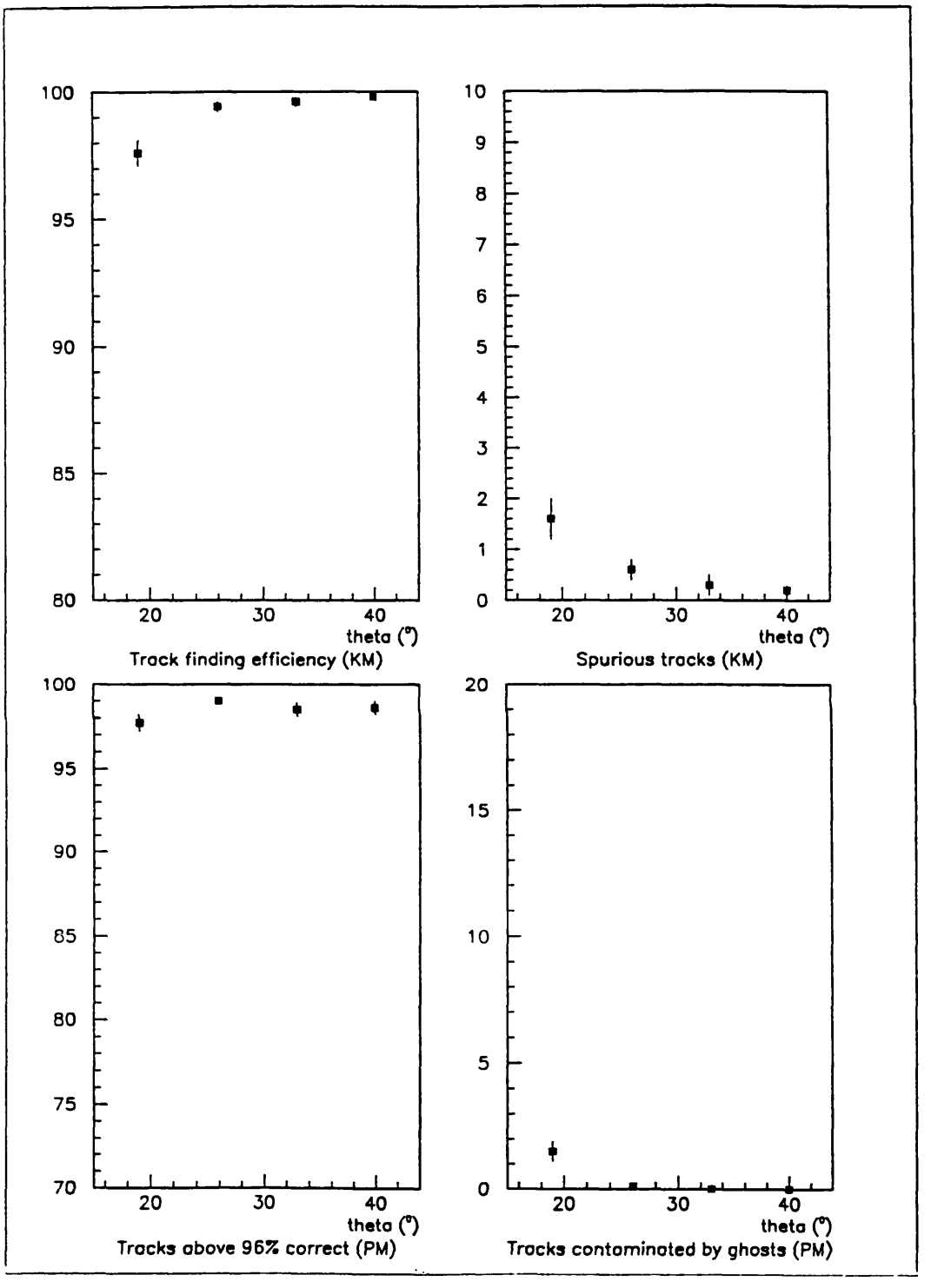

Figure 12.2: Single tracks inclined fixed values of θ.

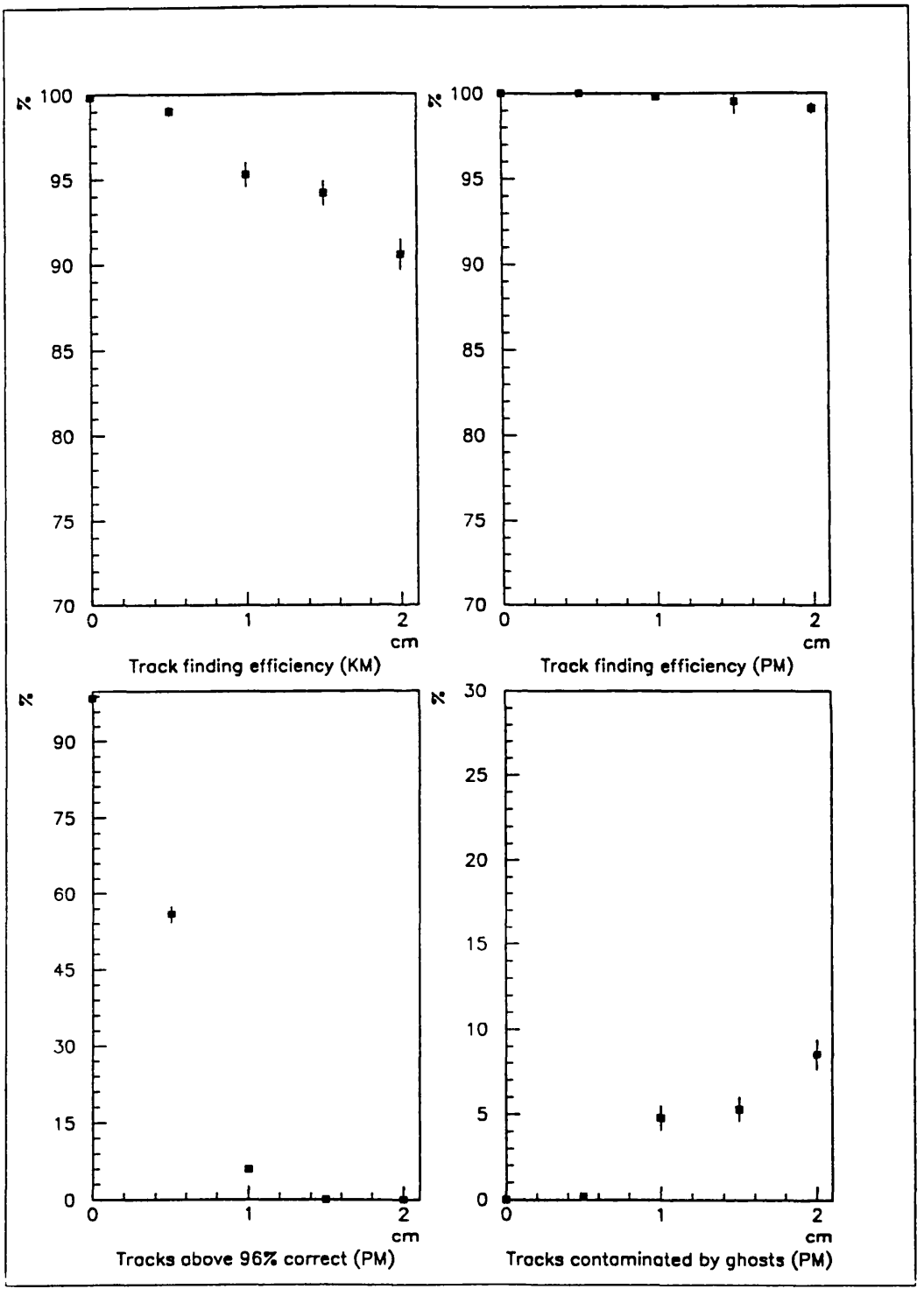

Figure 12.3: Single tracks with a displaced vertex.

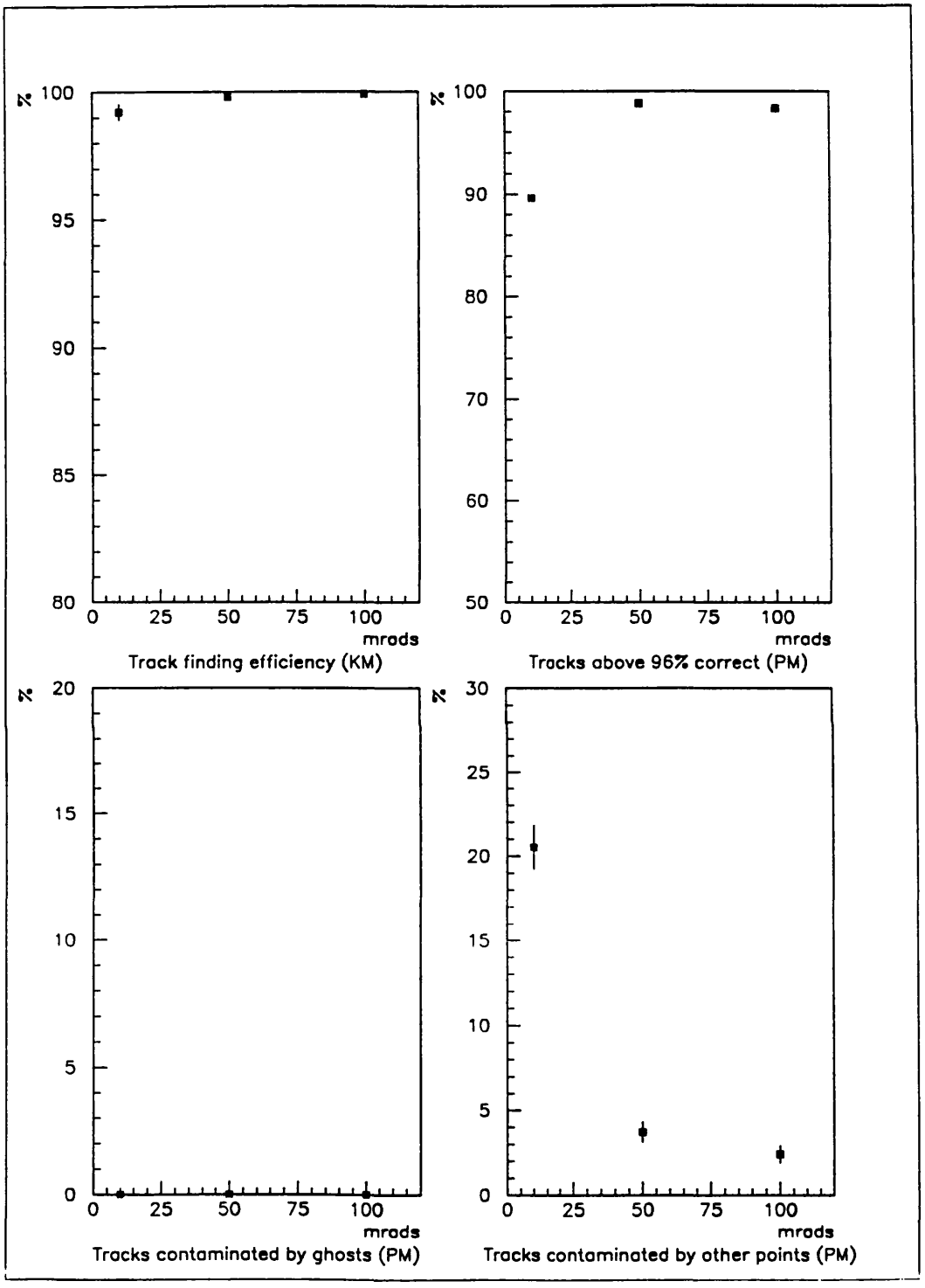

Figure 12.4: Two tracks separated by a small angle.

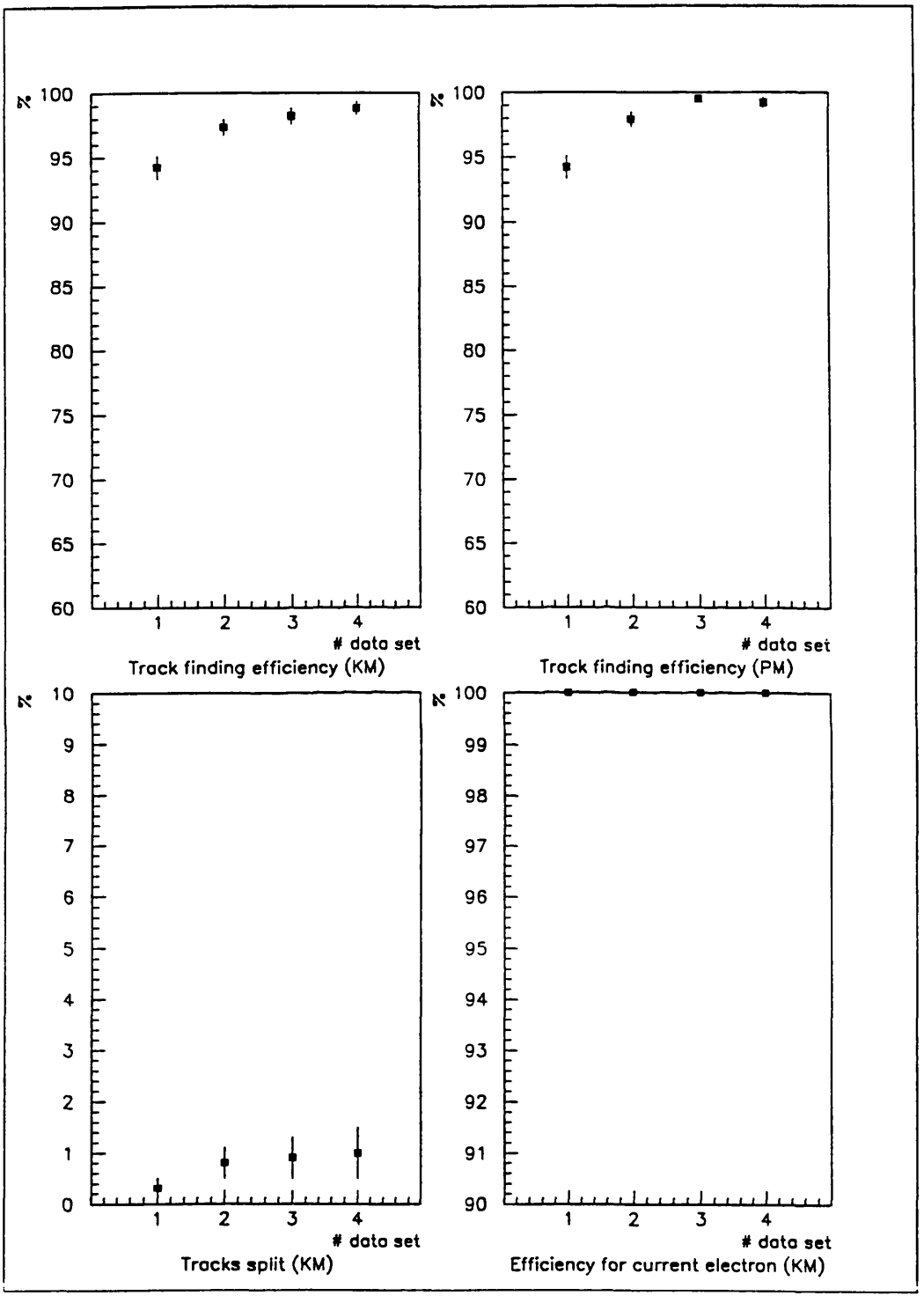

Figure 12.5: Neutral current events at characteristic points in the HERA kinematic plane: efficiency factors.

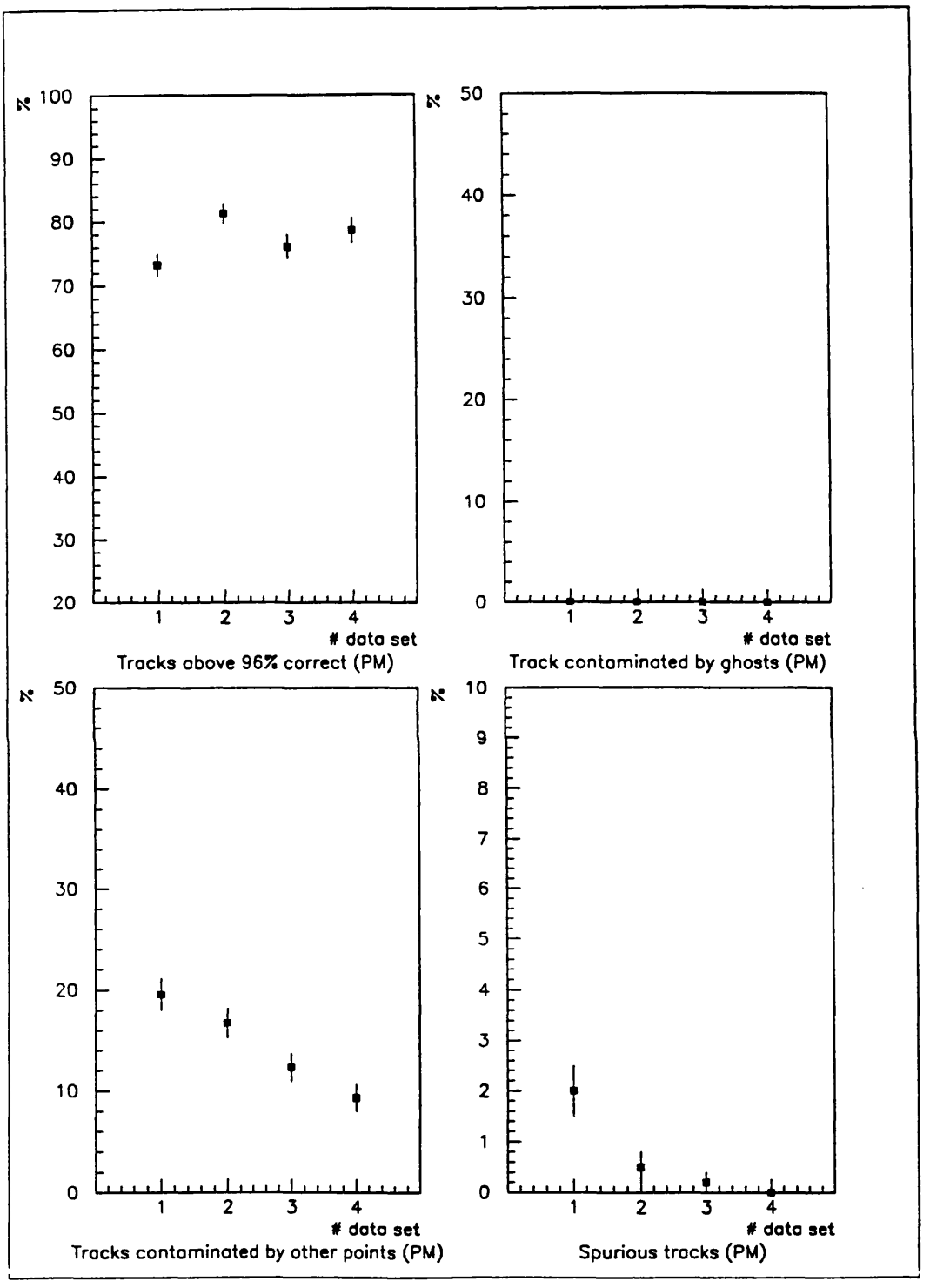

Figure 12.6: Neutral current events at characteristic points in the HERA kinematic plane: quality factors.

a relatively small inclination to the proton beam direction. These jets can contain short tracks that are very close together, some of which will lie within the two hit resolution of the chamber for a significant part of their trajectory. To investigate the effect of this on track finding, we introduce an extra fiducial cut in our analysis procedure. We exclude all Monte Carlo tracks that have an initial direction (ψ_0) within 15mrads of another Monte Carlo track. Tracks closer than this are prone to overlap with each other.

Figure 12.7 has been prepared using the following fiducial cuts:

1. $P_t > 0.45$ GeV/c
2. $19° < \theta < 161°$
3. $\Delta\psi_0 > 15$mrads

As can be seen, the track finding efficiencies calculated by both the kinematic matching and point matching analyses are at around the 99% level for all the NC data sets. We conclude that even in the most demanding NC events we find virtually all of the Monte Carlo tracks which are within our geometric acceptance.

12.4 Vectorisation

	ms
Scalar mode	3.57
Vector mode	1.97

Table 12.1: One seed finding attempt by the conformal seeder on an IBM 3090.

The Conformal seed finder has the advantage that it is highly suitable for implementation on machines with vector architectures.

We have completed a trial in which the core of the Conformal seed finder was implemented on the IBM 3090 vector facility at the Rutherford Appleton Laboratory. This section of code takes a list of input points, forms the link histogram, and searches it for a peak. In our trial we looked for a signal of 40 points, from among a total of 1000 points. This corresponds to searching for one seed in a typical CTD event.

Operating in scalar mode it was found that 93% of CPU time was spent initialising and filling the link histogram. Happily this section of the code is highly vectorisable, its performance in scalar and vector modes is presented in table 12.1.

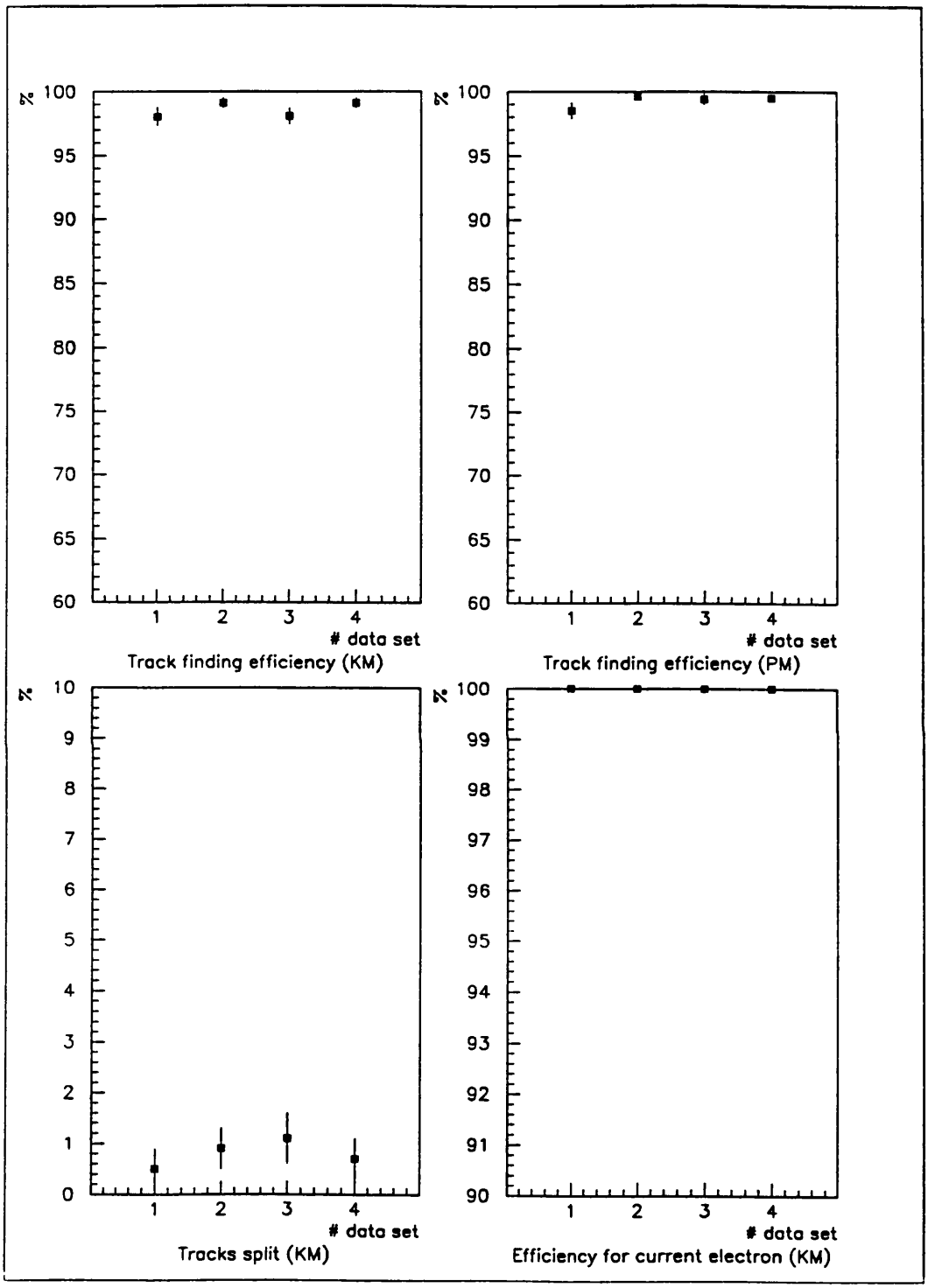

Figure 12.7: Neutral current events at characteristic points in the HERA kinematic plane: excluding tracks lost due to CTD two hit resolution.

12.5 Conclusions

This chapter has demonstrated the strength of the CTD track finding system.

- Excellent track finding performance is achieved over a wide region of the HERA kinematic plane.
- Useful opportunities exist to exploit vector architectures.

With confidence, we can recommend this track finding system for use in ZEUS physics analysis.

Bibliography

[1] D.Shaw & K.Long, ZEUS collaboration meeting, DESY, Sept 1990.

[2] D.H.Saxon, NIM A234 (1985) 258-266.

Chapter 13

An Application to Physics

[1]To illustrate the power of the CTD 2D track finding system we shall investigate its use in a possible ZEUS physics application. This speculates that during the early stages of HERA running the ZEUS calorimeter will not be sufficiently understood to allow the use of missing energy cuts during the separation of NC and CC events. In this chapter we shall

- Discuss techniques for separating NC and CC events.
- Present the separation results achieved using a novel technique based on the CTD 2D track finding system.

13.1 Separation of NC and CC Events

The separation of NC and CC events within the ZEUS detector has been discussed by [2]. This separation technique identifies CC events by inferring the presence in the event of a high energy neutrino. To do this, we measure $\left|\sum \vec{P}_t\right|$, the vector sum of all the transverse momentum observed in the calorimeter. In the ideal case this is zero in NC events, and equal to the transverse momentum carried away by the unobserved neutrino in CC events.

The finite resolution of the calorimeter means that this simple technique has to be augmented by placing a cut on the total energy observed in the calorimeter and, in the low-x region, by placing a cut on the isolation of an electron candidate

In [2] it was shown that NC and CC event sets with $Q^2 > 1000$ GeV^2 could be efficiently separated from each other using this technique. However, two assumptions are made:

1. The calorimeter is effectively hermetic.

[1]This study was first presented in [1]

2. Its calibration is well understood.

Both of these assumptions will eventually be valid at ZEUS, but during early running, this may not be the case.

13.2 Separation using 2D Tracks

We proceed on the assumption that we must find a way to separate NC and CC events based on tracking information. This might be necessary to complement imperfect information available from the calorimeter. The performance achieved by this method will be a useful demonstration of the power of the CTD 2D track finding system.

From our list of 2D reconstructed tracks, we discard all those not consistent with coming from the nominal interaction point. For the remaining tracks we use this assumed point as a constraint, and calculate two quantities:

$\mathbf{P_t}$ The apparent transverse momentum of the track.

ψ_0 The apparent direction of the track at the nominal interaction point.

We now discard all tracks with a P_t less than some suitable value. From the n tracks remaining we define the *span* (S) of an event, by summing over all track pairs,

$$S \equiv \frac{\sum_{ij}(\psi_{0i} - \psi_{0j})^2}{(n-1)}$$

Only events with $n \geq 3$ are considered.

Defined thus, the value of span reflects the differing event topologies: CC events containing only one jet will have a low span, NC events will have a higher span due to the contributions associated with the current electron, while photoproduction events containing jets balanced in transverse momentum will tend to have a still higher span. We can use span as a tool with which to help separate these event classes.

13.3 Preliminary Study

As a test of our method we shall try to isolate a sample of CC events in the region $Q^2 > 1000$ GeV^2.

For a full study of the this problem we would require a simulation of the ZEUS trigger system. Unfortunately this is not yet available within the ZEPHYR framework. We shall proceed making the reasonable assumption that all deep inelastic events with $Q^2 > 100$ GeV^2 are accepted by the trigger system. Certain types of photoproduction event will also pass the trigger, and in rare cases these

may appear similar to CC events. In [3] it was shown that in these cases the apparent Q^2 is usually less than 500 GeV^2.

We consider three possible backgrounds to our CC $Q^2 > 1000$ GeV^2 event sample:

1. NC events.
2. Photoproduction events.
3. CC events with $Q^2 < 1000$ GeV^2.

Contamination sources 2 and 3 will produce events with a real or apparent Q^2 well below that of the kinematic range in which we are interested. In the rest of this discussion we shall assume that these events can be excluded based on a Q^2 measurement made with whatever residual calorimeter information is available. Even if this were not the case, we could argue that most photoproduction events would have a very high span, while only 1/3 of the CC cross section has $Q^2 < 1000$ GeV^2. To investigate any possible contamination arising from NC events we will test our system using simulated data.

In chapter 12 we measured the performance of the 2D track finding system using only idealised data. This assumed a constant axial magnetic field throughout the CTD and did not include any simulation of physics processes such as multiple scattering and energy loss. To proceed with our study we must use the fullest possible simulation of the CTD. Using MOZART (see appendix B), we have simulated events in the CTD to include the following effects:

- All physics processes including multiple scattering, energy loss and particle decays.
- A realistic CTD magnetic field.
- The finite extent of the interaction region.
- Position resolution.
- Two hit resolution.
- Signal propagation delays and time of flight corrections.

This is the fullest simulation of the CTD currently available.

Using these conditions we generated 1000 NC events with $Q^2 > 100$ GeV^2 (simulating the NC events passing the trigger), and 200 CC events[2] with $Q^2 > 1000$ GeV^2 (simulating the CC events we wish to recover). These events were then processed within ZEPHYR. The following gardening algorithm was used (see chapter 11): use the conformal seeder to find seeds until only points in

[2]Statistically fewer events are needed in the CC sample because of its smaller cross section.

superlayer 1 remain, then use the segment seeder. The cut $P_t > 0.5$ GeV/c was used when forming the span value.

The present version of the CTD track finding system is not yet optimised for use with this fully simulated data. Such optimisation will not be done until the full design described in chapter 11 is implemented within the ZEPHYR framework. However, given the general robustness of our approach we are not surprised to find that even this non-optimised tracking system is adequate for the present study.

Figure 13.1 shows the value of span for NC and CC events assuming the perfect reconstruction of all Monte Carlo tracks. Figure 13.2 plots the same quantities obtained using the actual reconstructed tracks. As can be seen the separation is very good in both cases, with no NC contamination at values of span less than 5. Some entries develop anomalously large values of span. For CC events this is thought to be caused by the presence of gluon jets, while for NC events a great number of large entries result because the track corresponding to the scattered electron is split during track finding.

Using our reconstructed tracks, we find that 67% and 85% respectively of our NC and CC samples had $n \geq 3$, and so could be separated using this technique. Of these CC events, 83% were in the span region with no NC contamination.

Our NC and CC event samples have associated cross sections of 5300pb, and 44pb respectively. We have thus shown that $\sim$ 31pb of our CC sample is recoverable, in a region where the NC contamination is less than 6pb.

To support this study, we have repeated these measurements using CC data sets with $Q^2 > 100$ GeV^2 and $Q^2 > 500$ GeV^2. Figure 13.3 shows the fraction of the cross section in the each of these categories that falls in the region with no NC contamination. If this work is to be taken any further, a more detailed investigation of the distribution of our signal events in the xQ^2 plane should be made.

13.4 Conclusions

This preliminary study encourages us to believe that tracking information is a powerful tool for the classification of HERA events. Even in the absence of calorimetry, the NC contamination of a high-Q^2 CC event sample can be kept to an acceptable level. This speculative physics application demonstrates the power of the CTD 2D track finding system.

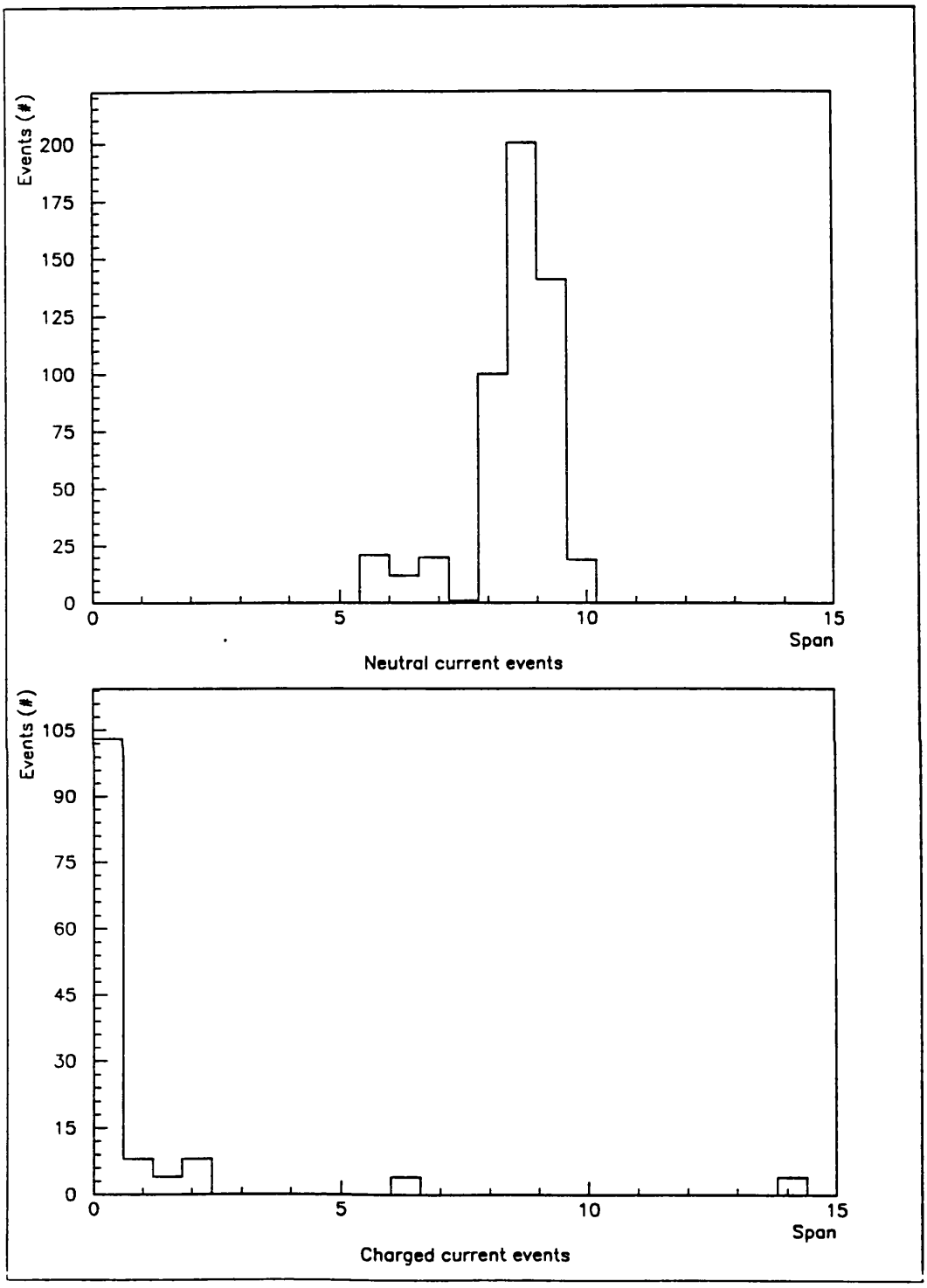

Figure 13.1: Span distributions for NC and CC events: using Monte Carlo tracks.

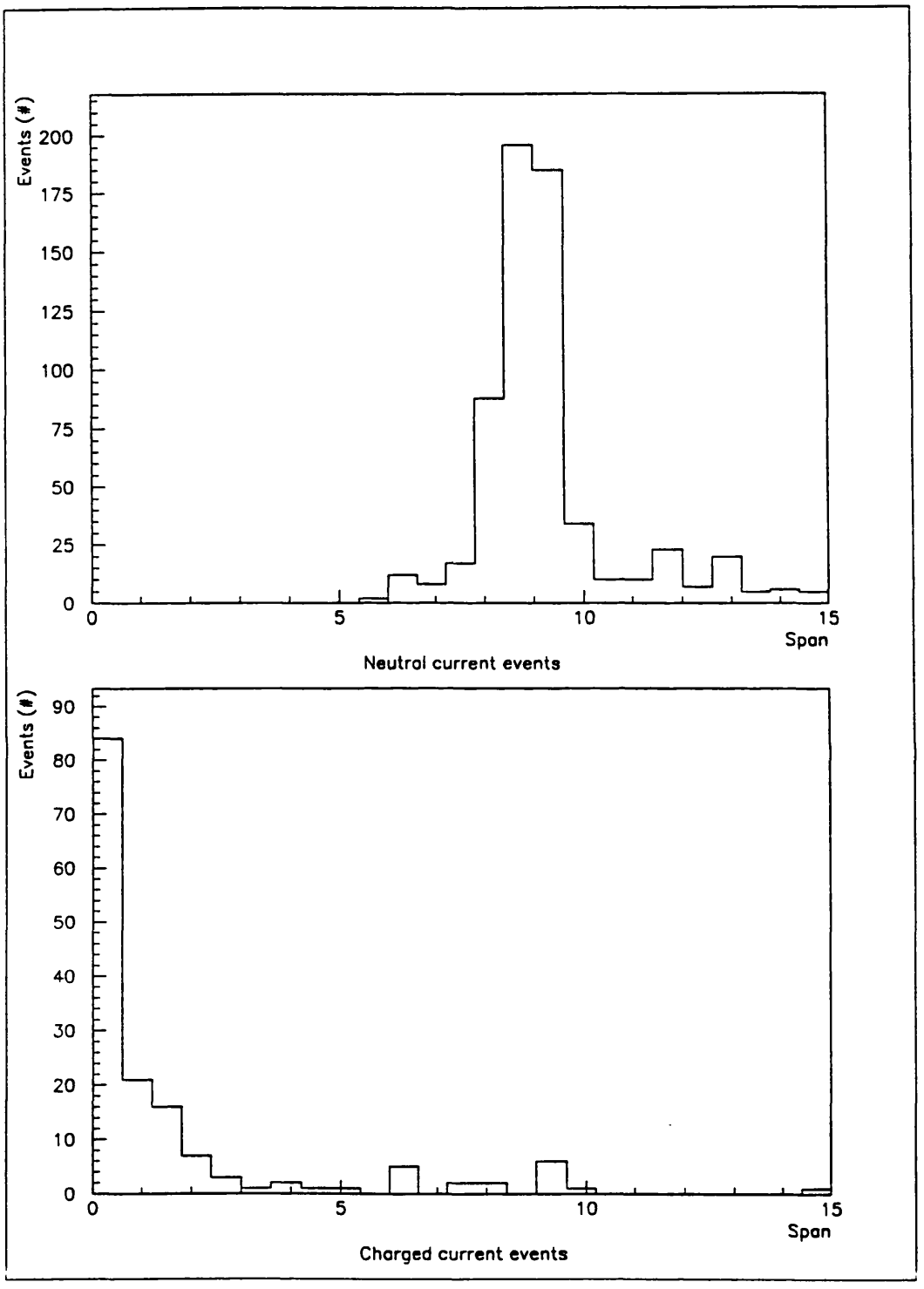

Figure 13.2: Span distributions for NC and CC events: using reconstructed tracks.

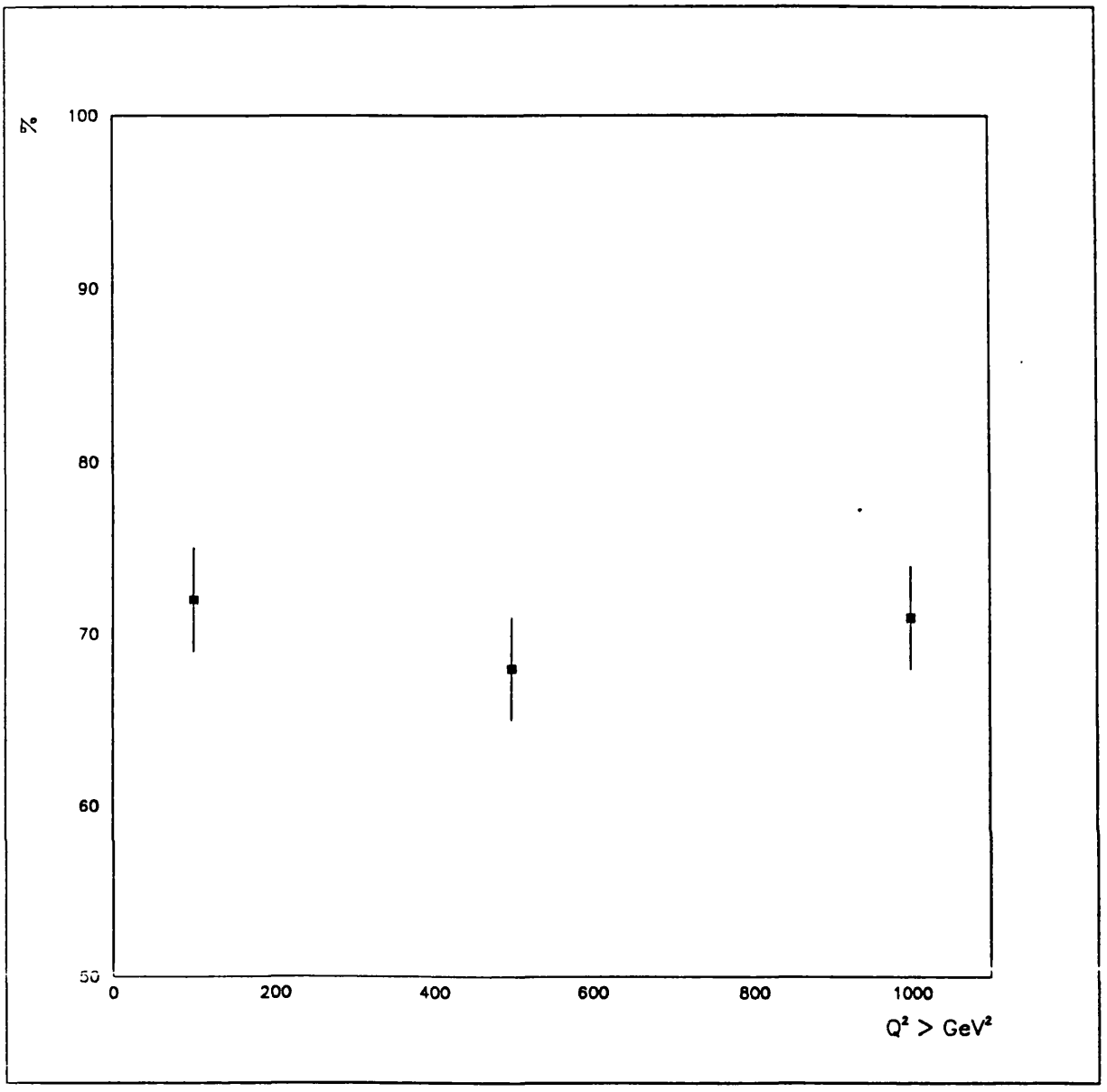

Figure 13.3: The fraction of CC events found in the region with no NC contamination.

Bibliography

[1] D.Shaw, ZEUS collaboration meeting, DESY, Sept 1990.

[2] G.Ingelman et al, 'Separation of Deep Inelastic Charged and Neutral Current Events' HERA workshop vol.1 pg.19.

[3] S.J. de Jong 'Hard Scattering of (almost) Real Photons at HERA' HERA workshop vol.2 pg.533.

Part IV

Appendices

Appendix A

Analysing pattern recognition performance

[1]Within high energy physics, no standard methodology for evaluating the pattern recognition performance of track finding algorithms has emerged. A typical approach starts by comparing the reconstructed tracks with a knowledge of the Monte Carlo simulated input. Unfortunately this comparison can become complicated and is often not well defined.

In this appendix we present a systematic solution to this problem, designed for use within cylindrical tracking chambers. Our technique allows a detailed insight into the performance of an algorithm and can be applied to any number of events. It has been successfully used to test and develop the online and offline algorithms discussed in this thesis.

A.1 Overview of Analysis Technique

Our evaluation technique begins by making use of Monte Carlo simulated data. To access the performance of a track finding algorithm each reconstructed track (or track segment) must first be matched to a Monte Carlo track. Once this assignment has been made the quality of these reconstructed tracks can be assessed.

The complication of the left-right hit ambiguity leads to the definition of two track *grids*. A track grid is a two dimensional array where the points placed on a reconstructed track (i) are cross-referenced with the Monte Carlo tracks (j) from which they came. One grid is defined such that it contains all the points (genuine and ghost) placed on reconstructed tracks ($grid_{All}(i,j)$). The other grid contains only genuine points placed on reconstructed tracks ($grid_{LR}(i,j)$).

[1]This methodology was first presented in [1]

A.1.1 Track Assignment

If all reconstruction were perfect then track assignment would be trivial. However, inefficiencies and the contamination of good tracks with incorrect points leads to problems. We suggest the use of two parallel approaches for assignment:

- Assignment based on track kinematics (kinematic matching)
- Assignment based on points (point matching).

One strength of our evaluation technique is the two different approaches used for matching, and the different perspectives they provide.

Kinematic matching is based on comparing the kinematic parameters obtained from the reconstructed tracks with the kinematic parameters of the Monte Carlo tracks. Using the estimated measurement error a χ^2-value can be calculated. A reconstructed track is assigned to a Monte Carlo track if this χ^2 is small.

Point matching is based on comparing entries in track grids. For each reconstructed track the maximum entry in $grid_{LR}$ is found, this is called the *leading entry*. The *contamination* of a track is then defined as the total number of points placed on the track which are not in the leading entry. A reconstructed track is assigned to a Monte Carlo track if it passes a cut on the ratio of the contamination to the leading entry.

In both approaches any sensitivity to the precise value of the cut should be investigated.

After assignment using either approach, fiducial cuts may be placed on the sample of Monte Carlo tracks that are considered. These cuts ensure that only Monte Carlo tracks in a specified kinematic range are counted against efficiency. The track dip angle and the track transverse momentum are good parameters to be restricted for cylindrical drift chambers in axial magnetic fields.

To access the general performance of an algorithm we define the following track classes (see figure A.1).

- If a Monte Carlo track fails the fiducial cuts it is *nasty*, otherwise it is *good*.
- If a good track has no reconstructed track assigned to it is *lost*, otherwise it is *found*.
- If a found track has several tracks assigned to it, it is *split*.
- If a reconstructed track is not assigned to any Monte Carlo track it is *spurious*.

Once the assignments have been made counters are incremented for the relevant track classes.

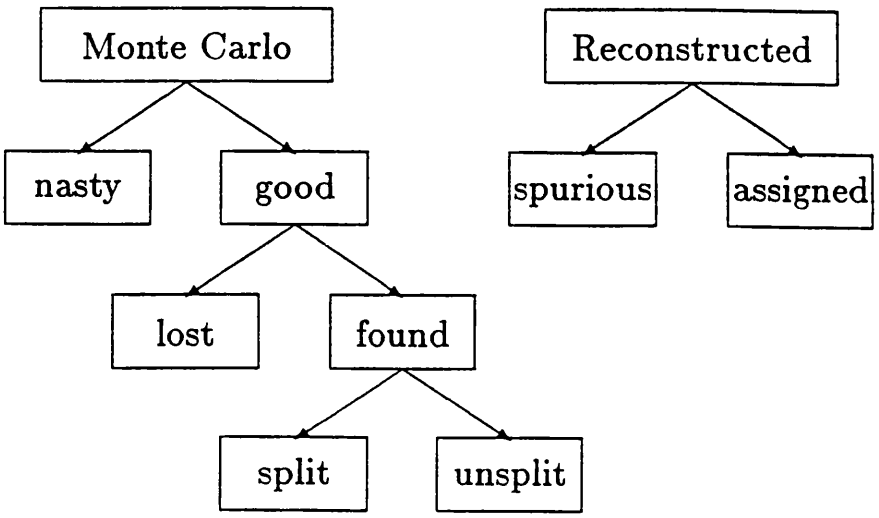

Figure A.1: The classification of Monte Carlo and reconstructed tracks.

A.1.2 Track Quality

After reconstructed tracks have been assigned they can be assessed for quality. For reconstructed track i, assigned to Monte Carlo track j, the following variables are defined:

Monte Carlo Total: The true number of points on Monte Carlo track j.

Correct Entry: The total number of points correctly placed on reconstructed track i, which has been assigned to Monte Carlo track j ($grid_{LR}(i,j)$).

Ghost Entry: The total number of points from the correct Monte Carlo track j, placed on the reconstructed track i but with the wrong sense of the left-right ambiguity ($grid_{All}(i,j) - grid_{LR}(i,j)$).

Wrong Entry: The total number of points from different Monte Carlo tracks, $k \neq j$, placed on reconstructed track i ($\sum_{k \neq j} grid_{LR}(i,k)$).

These variables can then be used to form quality factors which may be histogrammed for later analysis. These can typically by summarised by three important values:

1. The percentage of found tracks whose correct entry is greater than a given fraction of the Monte Carlo total for that track.

2. The percentage of found tracks with a non-zero ghost entry.

3. The percentage of found tracks with a non-zero wrong entry.

A.2 Applications

The above evaluation technique has been successfully used throughout this thesis. The same methodology has been applied to the study of both the online and offline applications.

A direct example of the comparative power of our method can be seen in chapter 10. In this study, two pattern recognition algorithms were being compared for offline applications. The conformal mapping algorithm assumed that all tracks came from the beam-crossing point, while the segment matching algorithm did not make this assumption. Figure 10.3 illustrates the results obtained for five different data samples of single track events. Each data sample contained 1000 tracks displaced from the origin by a fixed amount, which differed for each data set.

The graph entitled 'Track finding efficiency (KM)' was made using the kinematic matching approach and shows the track finding efficiency of the conformal mapping algorithm falling as expected. However, the graph entitled 'Track finding efficiency (PM)', which was made using the point matching approach, shows that both algorithms maintain a roughly constant track finding efficiency. This information, combined with that available from the quality factors discussed above, allowed us not only to observe the performance of the conformal mapping algorithm, but also to understand the exact way in which it was occurring. This useful information might have been missed by a simpler and less systematic method of analysis.

In the offline studies both algorithms were designed to find tracks comprising of up to 40 points. However, in the studies of the online segment finding algorithm the typical track length was only 5 points (see chapter 6). In both these contrasting examples it was found that the same basic evaluation approach could be applied and gave a good insight into the algorithm performance.

A.3 Conclusions

We have developed a systematic technique for evaluating the performance of pattern recognition algorithms used with tracking chambers. A strength of our approach lies in the use of two complementary techniques for assignment. The method has been employed successfully throughout this thesis.

A.4 Glossary of Analysis Terms

MC track: A track generated by a Monte Carlo simulation program.

Reconstructed track: A track found by the reconstruction algorithm under study.

Fiducial region: The region of track kinematic space in which we which to test our pattern recognition performance.

Good track: A MC track lying inside the fiducial region.

Nasty track: A MC track lying outside the fiducial region.

Found track: A good MC track that has a reconstructed track assigned to it.

Lost track: A good MC track that does not have a reconstructed track assigned to it.

Split track: A good MC track that has more than one reconstructed track assigned to it.

Spurious track: A reconstructed track that is not assigned to any MC track.

Genuine Points: Points formed from the correct sense of the left-right ambiguity.

Ghost Points: Points formed from the wrong sense of of the left-right ambiguity.

All Points Grid: A grid used to cross-reference all points placed on reconstructed tracks with their parent MC track.

Genuine Points Grid: A grid used to cross-reference genuine points placed on reconstructed tracks with their parent MC track.

Entry: Each element of either grid is termed an entry.

Leading Entry: The maximum entry in the genuine points grid.

Contamination: The total number of points placed on a reconstructed track minus the number of leading entry points.

Point matching: Track assignment based on point grids.

Kinematic matching: Track assignment based on reconstructed and MC track kinematic values.

Bibliography

[1] D.Shaw, K.Long & D.Gingrich 'A Technique for Evaluating Pattern Recognition Performance in Tracking Chambers' Oxford preprint OUNP-90-2.

Appendix B

Monte Carlo simulation

B.1 Simulation of HERA Physics

For the purposes of this thesis deep inelastic physics processes have been simulated using LEPTO version 5.2 [1].

This program has been used to implement the leading order electroweak cross sections [2], with QCD corrections [3]. It uses the quark distribution functions given by parameterisation I in [4]. Hadronisation is performed using the LUND string model [5].

B.2 Simulation of the ZEUS Detector

Simulated ZEUS data can be generated using a variety of different Monte Carlo simulation programs. For this thesis, the two most important of these have been the ZEUS trigger Monte Carlo (ZG313T1) [6], and MOZART (T1ZGEN) [7]. Both of these programs are based on the GEANT package [8], and share many routines in common. In particular they share a common simulation of the CTD.

The simulation of the CTD uses the planar drift approximation (PDA) to assign hits and drift times. A CTD cell is partitioned into eight planar drift cells, one centred on each wire, and aligned to the nominal drift direction. A hit is recorded on a sense wire if a Monte Carlo track enters its planar drift cell. The drift time associated with this hit is taken from the point at which the track trajectory crosses the middle of the planar cell. This measurement is not immediately smeared to represent the CTD position resolution, but *is* modified by adding time of flight and propagation delay effects.

When a track only crosses the corner of a planar drift cell, the procedure outlined above can produce drift times that correspond to a point outside of the real cell. In such cases a more complex simulation will eventually be needed.

When Monte Carlo tracking data is simulated, several steps are normally required before it is ready to be passed to a reconstruction program. For the

CTD these steps include

1. Smearing hit drift times to represent CTD position resolution.
2. Deleting hits to simulate the CTD two hit resolution.
3. Adding hits to simulate noisy channels.
4. Deleting hits to simulate inefficient channels.

These steps are carried out internally by MOZART while for the trigger Monte Carlo they are added by utilities in the ZGANA package [9]. ZGANA also provides a simulation of the ZEUS FLT.

B.3 Fast Simulation of the ZEUS Detector

The SIMPLE Monte Carlo [10] provides a fast simulation of the ZEUS detector. For applications where detail is not required, this Monte Carlo offers the greatest ease of use.

CTD position resolution and two hit resolution are simulated, but a constant axial magnetic field is used throughout the CTD region and physics processes such as energy loss and multiple scattering are neglected.

Bibliography

[1] G.Ingelman, LEPTO version 5.2, DESY.

[2] G.Ingelman 'Deep Inelastic Physics at HERA' DESY 87-144 (1987).

[3] M.Bengtsson et al, 'Parton Cascade Evolution and Event Structure at HERA' HERA workshop vol.1 pg.149.

[4] E.Eichten et al, Rev. Mod. Phys. 56 (1984) 579, ibid. 58 (1986) 1047.

[5] B.Andersson et al, Phys.Rep. 97 (1983) 31.
T.Sjöstrand, M.Bengtsson, Computer Phys. Comm. 43 (1987) 367.
T.Sjöstrand, Computer Phys. Comm. 39 (1986) 347.

[6] G.F.Hartner & I.Yoshihisa 'The Trigger Monte Carlo ZG311T6' Toronto, Canada.

[7] N.Mccubbin, Rutherford Appleton Lab., Private communications, July 1990.

[8] R.Brun et al. Data Handling Division, CERN.

[9] 'ZGANA 6 - A Software Library to Process Output from the ZG311T6 Trigger Monte Carlo Programme' The Zeus Trigger Monte Carlo Group.

[10] S.Lloyd & B.Foster, SIMPLE write-up (unpublished).

Appendix C

ARAXNE

C.1 Overview

ARAXNE was the main development framework for prototype CTD pattern recognition algorithms. It has been used extensively by both the online and offline groups, providing many useful facilities:

Data structures: Using the BOS memory management system a comprehensive set of data structures were defined. These allowed the user to handle raw data, tracks and Monte Carlo truth information conveniently within the program. Separate data structures allowed the user to access CTD geometry information.

Transformation utilities: The TCTRAN utility allowed the user to make transformations between the different CTD co-ordinate systems.

Fitting utilities: The TCRRPH utility provided a fast circle fit to a set of input points. This could be used to assist any prototype pattern recognition algorithm.

In addition to these internal features, ARAXNE was supported by interfaces to several other important programs:

Trigger Monte Carlo: A simulation of the ZEUS detector (see appendix B).

SIMPLE Monte Carlo: A fast simulation of the ZEUS detector (see appendix B).

CTD Event Display: A versatile CTD utility capable of displaying hits, segments and tracks as well as cell and superlayer geometry.

CARP: A program for assessing the performance of pattern recognition algorithms. This program implements the analysis concepts discussed in appendix A.

A complete overview of ARAXNE is given in [1].

C.2 SLT Test System

The interface between the ARAXNE framework and the SLT segment finding routines was formed by the DSREAD package [2]. This package was designed to reflect the CTD SLT processing architecture, allowing online segment finding algorithms to be tested in a realistic environment. Its structure allowed the actual segment finding algorithm to be downloaded and executed on a transputer while the rest of the ARAXNE program ran on the host computer.

Bibliography

[1] K.Long 'Some Formulae for Pattern Recognition in r, ϕ' ZEUS-RAL-88-3.

[2] D.Shaw, J.B.Lane 'DSREAD version 5.0 Documentation' ZEUS-UCL-89-0003.

Appendix D

ZEPHYR

The CTD track reconstruction software is an integral part of ZEPHYR, the ZEus PHYsics Reconstruction program.

D.1 Software Tools

To co-ordinate a project on the scale of ZEPHYR, it has been necessary to use several modern software engineering techniques.

ADAMO is a complete FORTRAN data system, offering many different services to the user [1]. It provides a precise data definition language (DDL) for defining the data structures to be used by a program, together with tools to document the data dictionary and check it for consistency. It generates the code necessary to support this structure in the final application and provides run-time tools with which to manipulate it.

ADAMO stores data in 'tables' *(all hit points or all tracks)*, each row of which corresponds to one 'entity' *(a single hit point or a single track)*. Any individual entity may have a 'relationship' to an entity in another table *(hit points lie on a track)*. 'Selectors' may be defined to create sub-sets of the entities in a table *(only hit points from superlayer 1)*.

SASD methodology has been discussed for use at HERA by [2]. The elements of this system that we have found most useful for the purposes of design and documentation are data flow diagrams, entity relationship diagrams, and structure charts. Unfortunately we lack a software tool capable of combining all the normal SASD functionality with the entity relationship representation used by the ADAMO system.

The use of SASD techniques and ADAMO has greatly facilitated the implementation of a robust reconstruction program capable of supporting many interchangeable modules.

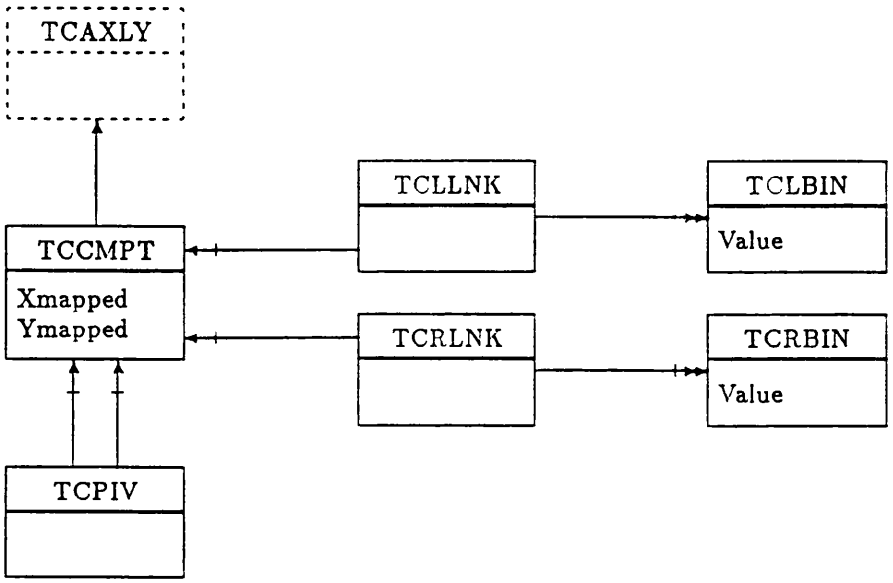

Figure D.1: The TCRECO data structure.

D.2 CTD Data Structures

The data structures within ZEPHYR are maintained, accessed and documented using the ADAMO data system.

Figure D.1 illustrates the main data structure used by the CTD 2D pattern recognition within ZEPHYR. Each track candidate entered in the TCCAND table is composed of many *bits*, each stored as an entity in the TCBIT table. In principle, the bits of which a track candidate is comprised could include different kinds of objects, such as track segments or points from stereo superlayers. However, all current algorithms associate each TCBIT entity with a hit point from an axial superlayer, represented by a relationship between each TCBIT entity and a TCAXLY entity. Hit points already entered in TCBIT will not be used in further seed finding attempts. The TCBBIT table acts as a further list of TCAXLY entries. This allows hit points to be excluded from further seed finding attempts without placing them on a track candidate. Each seed is represented by a TCSEED entity, which may or may not lead to a track candidate being entered in TCCAND.

Figure D.2 illustrates the main data structures used internally by the conformal seed finder. The TCCMPT table stores the co-ordinates in conformal space of each TCAXLY hit point. When a pivot hit is chosen an entry is added to the

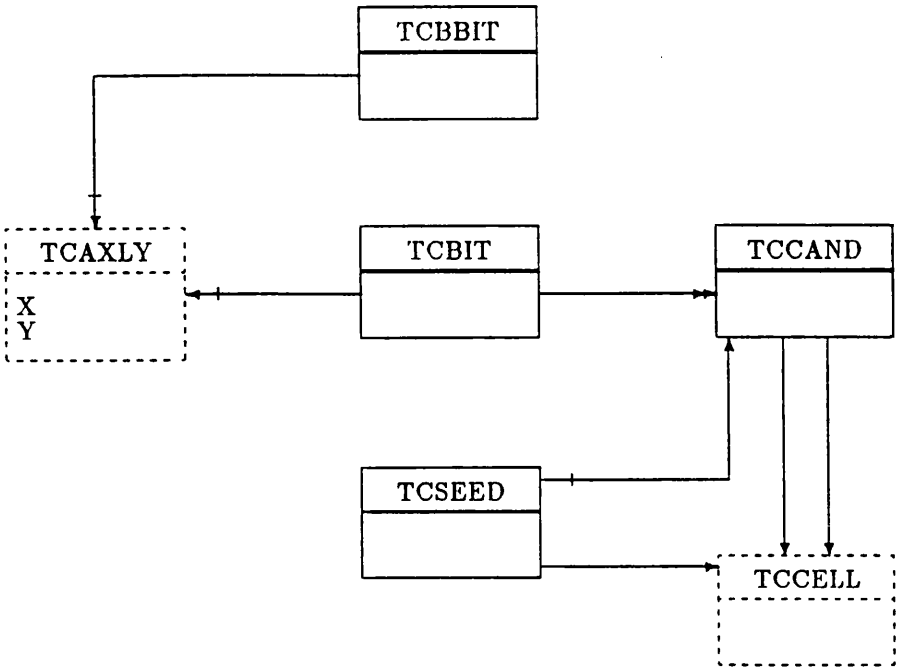

Figure D.2: The TCCONF data structure.

TCPIV table and relationships filled between it and its associated left and right pivot points in TCCMPT. The entities in TCLBIN and TCRBIN form the bins of the link histogram corresponding to the left and right pivot points respectively. The TCLLNK and TCRLNK tables act as 'switch-yards', indicating in which link histogram bin a hit point in TCCMPT falls (TCLLNK and TCRLNK are only used if the FitPeak option is selected).

The dataflows drawn in figures 11.1 to 11.6 of chapter 11 can be specified in terms of the ADAMO objects associated with these data structures:

Track candidates: The TCCAND and TCBIT tables.

Seed mask: The selector Sel_Seed_Mask on the TCAXLY table.

Axial hits: The TCAXLY table.

2D track candidate: An entry in the TCCAND table with its associated entries in the TCBIT table.

Bad point: An entry in the TCBBIT table.

Seed: An entry in the TCSEED table.

Hits used: Entries in the TCBIT and TCBBIT tables.

Pivot hit: An entry in the TCPIV table.

Current window: The selector Sel_TCCMPT_mask on the TCCMPT table.

Histograms: The TCLBIN, TCRBIN, TCLLNK and TCRLNK tables.

Peak: The selectors Sel_TCLBIN_Peak and Sel_TCRBIN_Peak on the TCLBIN and TCRBIN tables respectively.

Predicted kinematics: The PRED fields of the last TCSDEX entry, and all TCAXLY entries with a relationship to any TCSDEX entry.

Measured kinematics: The MEAS fields of the last TCSDEX entry, and all TCAXLY entries with a relationship to this TCSDEX entry.

Best kinematics: The BEST fields of the last TCSDEX entry, and all TCAXLY entries with a relationship to any TCSDEX entry.

D.3 Testing

The aim of this section is to demonstrate that the following packages are understood and working as specified:

MOZART: The ZEUS offline Monte Carlo program.

TCRRPH: The fast circle fit utility.

TCCMSD: The conformal seed finder.

TCSGSD: The segment seed finder.

TCEXSD: The seed extender.

To achieve this we will use TCANAL, the performance analysis package.

We saw in chapter 3 that 2D tracks may be specified by three kinematic parameters, $P_{Recon}(i)$, along with estimates of their covariance matrix, $C_{Recon}(i,j)$. Given that the measurement error on each hit point is gaussian, and in the absence of any pattern recognition mistakes, the form of our fitting procedure ensures that the reconstructed parameters will also be gaussianly distributed. We compare our reconstructed tracks to the true generated tracks, $P_{True}(i)$, by plotting two types of histogram:

Normalised gaussian distributions: We use our kinematic parameters for each track to plot $(P_{Recon}(i) - P_{True}(i)/\sqrt{C_{Recon}(i,i)})$ for $i = 1,3$. When plotted over a large number of tracks, the resulting distributions should be a gaussian with $\sigma = 1$ and mean $= 0$.

Uppertail probability of track fit: We form a χ^2 value for the association of P_{True} with P_{Recon}. The uppertail probability of this χ^2 should be distributed evenly between 0 and 1. When plotted over a large number of tracks, the resulting distribution should be flat with mean = 1/2.

We make use of data sets generated under the same conditions as those used in chapter 12. For our first test we use a NC data set with $Q^2 > 100$ GeV^2:

Test of MOZART: Figure D.3 demonstrates that the Monte Carlo data is understood. The distance from each true hit point to its associated Monte Carlo track is plotted. We expect that this histogram should be a gaussian with $\sigma = 130$ μm. This is what we observe.

For the next stage of testing we use single track data sets. We select the DropGhost option within ZEPHYR, which ensures that only hit points with the correct sense of the left-right ambiguity remain for processing by the seed finders. This data should be perfectly gaussianly distributed as it is not possible for a pattern recognition mistake to contaminate a track with an incorrect point:

Test of TCRRPH: Figure D.4 demonstrates that TCRRPH is working well for fits to as many as 40 or as few as 8 hit points.

Test of TCCMSD: Figure D.5 demonstrates that TCCMSD, the conformal seed finder is producing seeds with unbiased, correctly distributed parameters.

Test of TCSGSD: Figure D.6 demonstrates that TCSGSD, the segment seed finder is producing seeds with unbiased, correctly distributed parameters.

In our final test we de-select the DropGhost option and use TCCMSD and TCEXSD to process the NC data set:

Test of TCEXSD: Figure D.7 demonstrates that TCEXSD, the seed extender is producing tracks with unbiased, correctly distributed kinematic parameters.

We can happily conclude that all the major packages of the CTD 2D track finding system are working as expected.

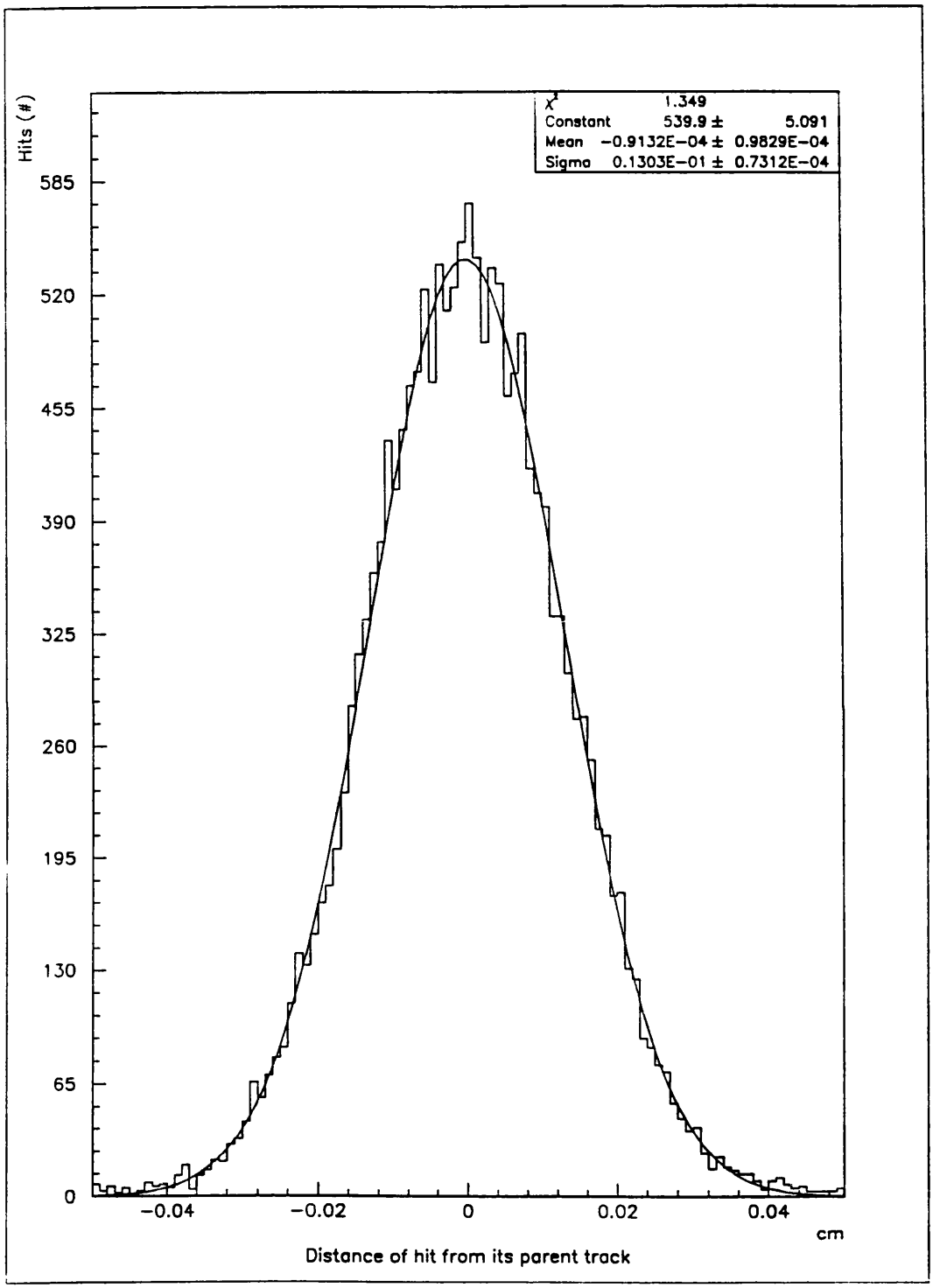

Figure D.3: Test of MOZART: the ZEUS offline Monte Carlo.

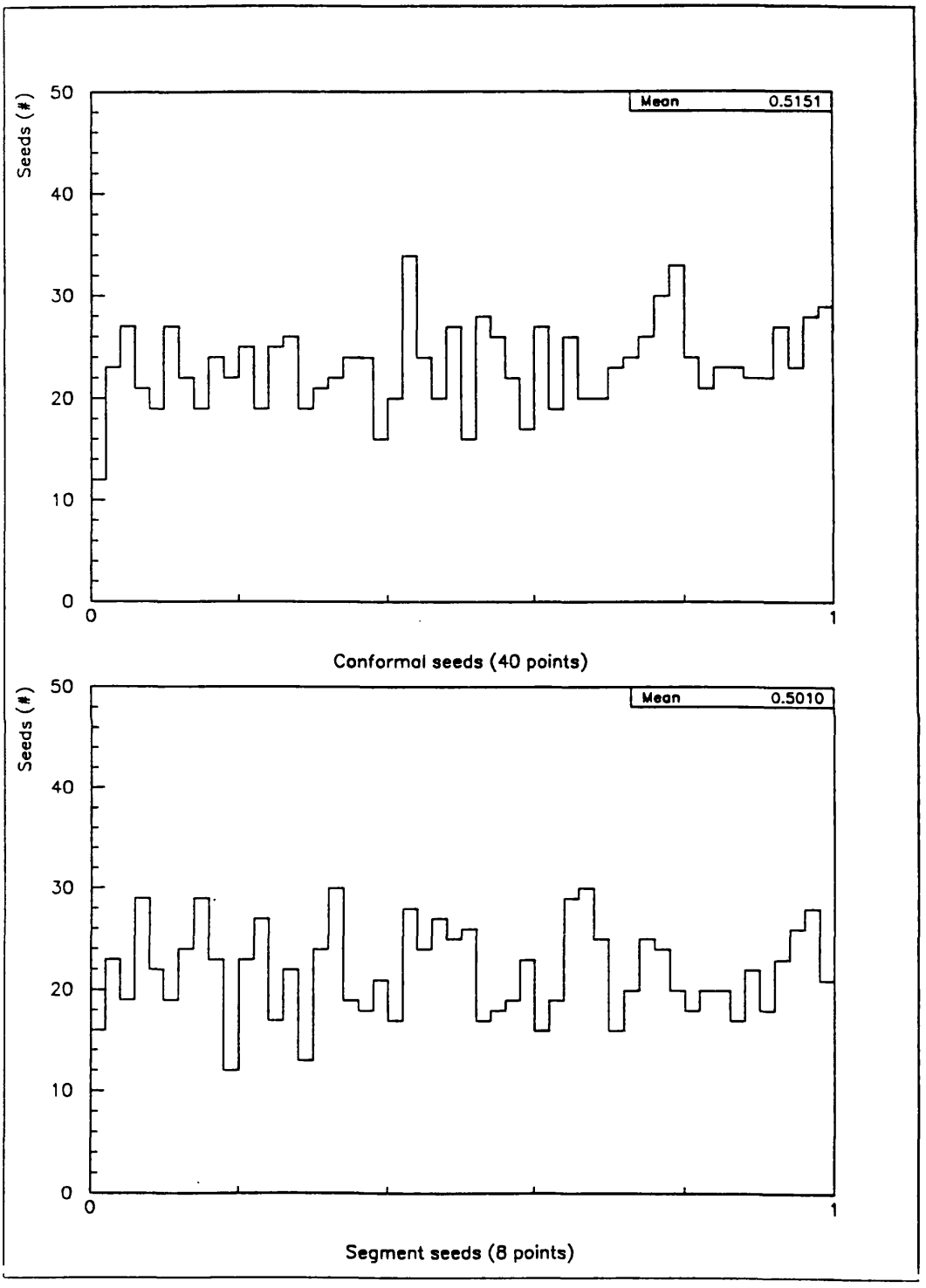

Figure D.4: Test of TCRRPH: uppertail probability of fitted χ^2.

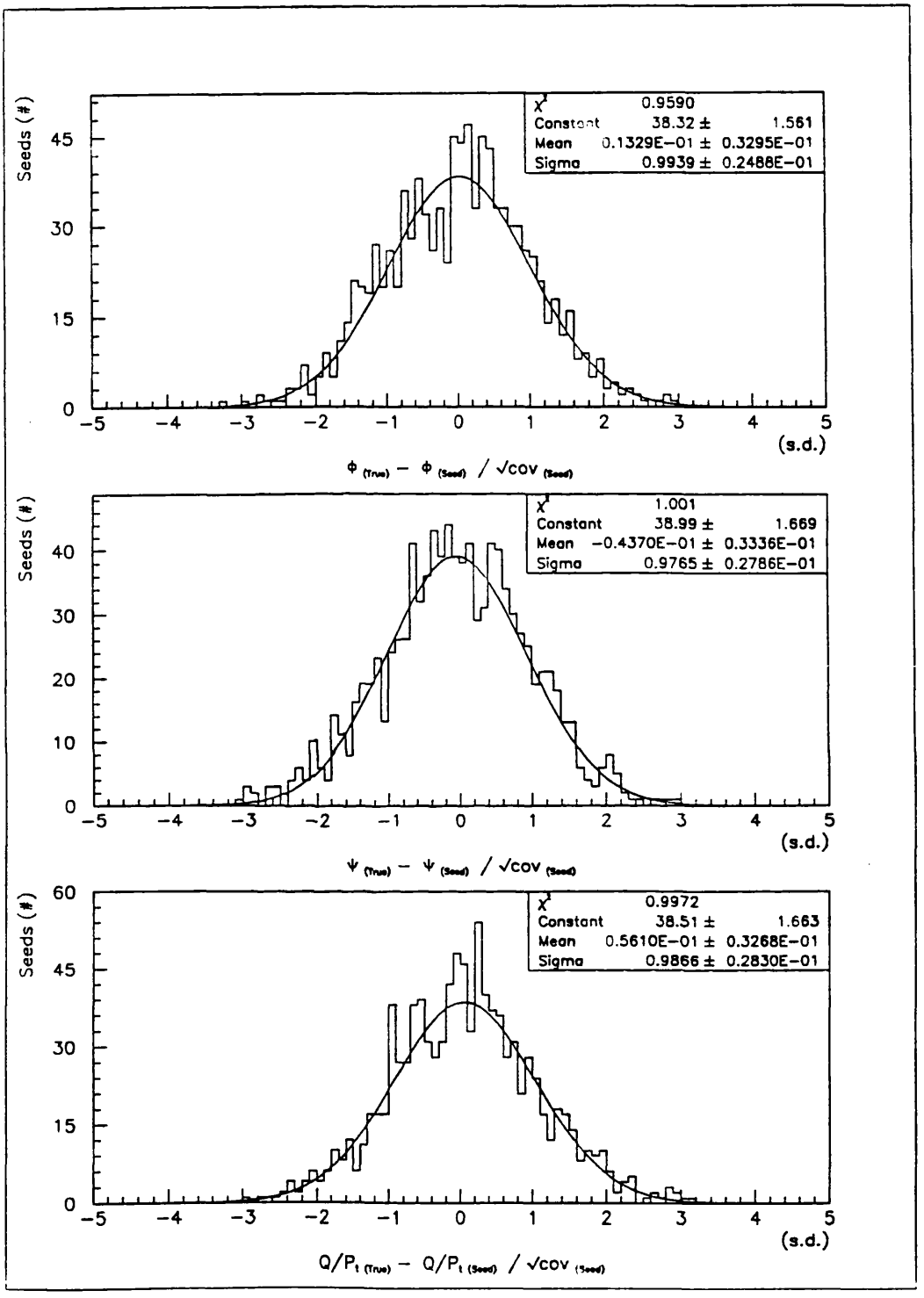

Figure D.5: Test of TCCMSD: normalised gaussians for the comparison of the seed kinematic parameters to the true kinematic parameters of the Monte Carlo track.

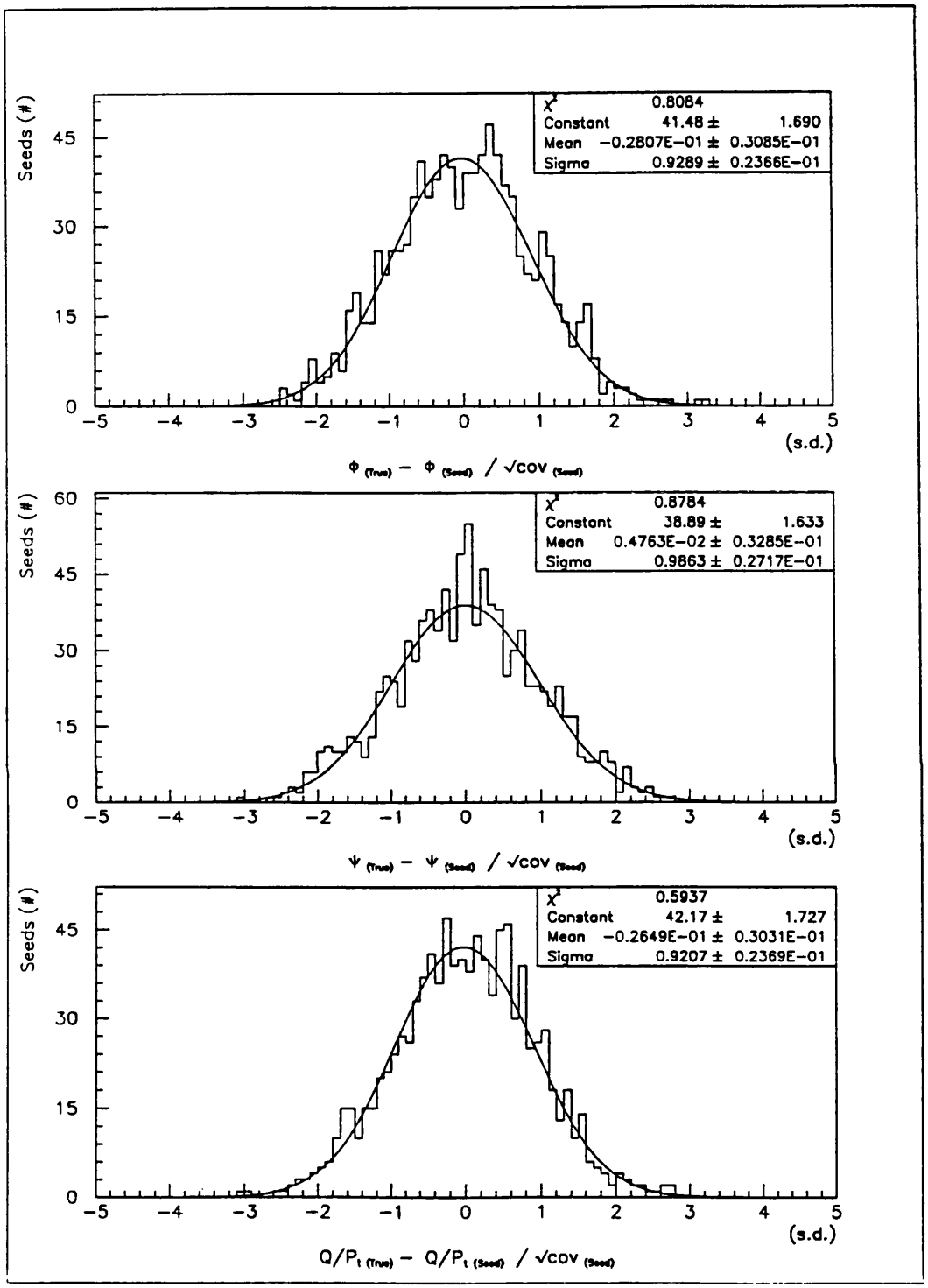

Figure D.6: Test of TCSGSD: normalised gaussians for the comparison of the seed kinematic parameters to the true kinematic parameters of the Monte Carlo track.

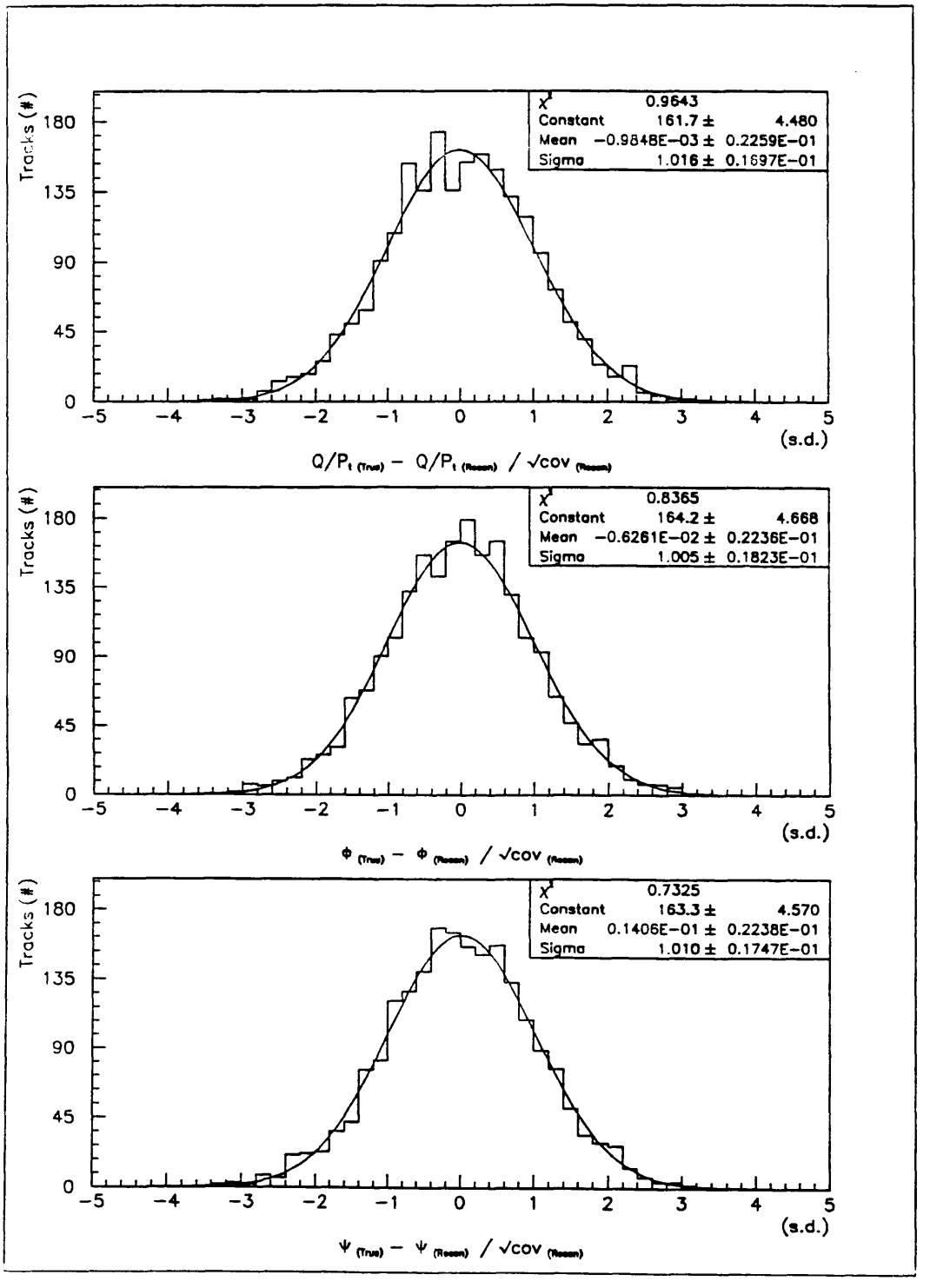

Figure D.7: Test of TCEXSD: normalised gaussians for the comparison of the reconstructed kinematic parameters to the true kinematic parameters of the Monte Carlo track.

Bibliography

[1] S.M.Fisher & P.Palazzi, Comput. Phys. Commun. 57 (1989) 169-175.

[2] K.S.Gather, Comput. Phys. Commun. 57 (1989) 29-36.