RecoilUtils
A ROOT based Analysis Framework for the HERMES Recoil Detector
Andreas Mussgiller
II. Physikalisches Institut
Friedrich-Alexander-Universität
Erlangen-Nürnberg
HERMES
DESY IT Seminar, 26/11/07
Outline

- The HERMES Recoil Detector
- General Idea
- A brief Project Overview
- Selected Features
  - uDST Framework
  - Cut Management
  - Event Display
- Summary and Future Plans
• Deeply virtual Compton scattering
• Recoiling proton undetected
• Process identified via missing mass
• About 15% background
DVCS at HERMES with Recoil Detector

- Recoiling proton detected
- Reduces background to about 1%
The HERMES Recoil Detector

- **Beam**

- **Target-Cell**

- **1 Tesla Superconducting Solenoid**

- **Photon Detector**
  - 3 Layers of Tungsten/Scintillator
  - PID for p > 0.5 GeV/c

- **Scintillating Fiber Detector**
  - 2 Barrels
  - 2 Parallel and 2 Stereo Layers in each Barrel
  - Momentum Reconstruction & PID

- **Silicon Detector**
  - 2 Layers inside Vacuum
  - 16 double-sided Sensors
  - 128 x 128 strips
  - Momentum Reconstruction & PID
Motivation

- Installed December 2005
- Start of Data taking February 2006
- Commissioning of detector components
  - Fiber tracker finished in February 2006
  - Silicon detector finished in September 2006

Understand the detectors:

- Noise
- Common mode
- Correlations
- ...
- Lots of plots to produce
- Many ideas

A flexible and modular analysis framework

- that uses many nice ROOT features
- not too complicated
Outline

- The HERMES Recoil Detector
- General Idea
- A brief Project Overview
- Selected Features
  - uDST Framework
  - Cut Management
  - Event Display
- Summary and Future Plans
HERMES Data Production Chain

Raw Data

HDC
HERMES DECODER

HRC
HERMES RECONSTRUCTION

XTC
EXTERNAL TRACK RECONSTRUCTION

HMC
HERMES MONTE CARLO

Slow Control

uDSTwriter
HERMES Data Production Chain

- Stored in ADAMO format
- Contains
  - Raw Data
  - Clusters
  - Spacepoints
  - Recoil Tracks
  - HERMES Tracks
- Used for detector studies

- Stored in ADAMO format
- Contains only reduced data
  - Clusters
  - Tracks
  - Slow Control
- Used for physics analysis
A Bit of ADAMO

- Based on Entity-Relationship Model
  - Entities with fixed number of attributes
  - Relationships between entities
The General Idea

• Each ADAMO table represented by a class
  – rdTrack \rightarrow TrdTrack
  – rdSpacePoint \rightarrow TrdSpacePoint

• Relationships between tables handled by pointers, TRef and TRefArray
  – All ADAMO navigations done in common code (ADAMO to ROOT interface)
  – Let ROOT do the rest

• Analysis done via hierarchy of *Analysis Modules*
  – Common modules for standard tasks (raw histos etc)
  – A specific analysis module for each dedicated study
  – Modules can be used by everyone
The General Idea

- A Track knows from which spacepoints it is made of
- A Spacepoint knows which tracks it belongs to
- Member functions provide “navigation”
  - From track to spacepoints
  - From spacepoint to tracks
The General Idea

**XTCA**

**XTCSelector**

**XTC**

EXTERNAL TRACK RECONSTRUCTION

ADAMO to ROOT Interface

Recoil Analysis Object

ROOT Files

ROOT Tree
The Recoil Analysis Object

- Handles processing of *Analysis Modules* (*RecoilAnalyzer*)
- Provides access to data (tracks, clusters, ...)
- Takes care of output
  - One output file with histos and cuts in directory structure
  - Optional second file with a ROOT Tree
  - Common interface to tree

```cpp
void AnaInit()
{
    Recoil::Get("DemoAnalysis");
    gRecoil->RegisterAnalyzer("ClassName", "Name", "Option");
gRecoil->RegisterAnalyzer("SomeOtherClass", "Name", "Option");
... 
gRecoil->RegisterAnalyzer("Filename.C", "OtherName", "Option");
}
```

- Allows multiple *Analysis Modules* of the same type
- Modules identified via class name and name
- Option string can be used to control module behavior
- Modules can be compiled at run-time
Eventloop

- Recoil::Begin()
  - RecoilAnalyzer::Begin()
  
  Called once at startup

- Recoil::BeginOfRun()
  - RecoilAnalyzer::BeginOfRun()
  
  Called at beginning of each HERMES run

- Recoil::Process()
  - RecoilAnalyzer::Process()
  
  Called for each event

- Recoil::EndOfRun()
  - RecoilAnalyzer::EndOfRun()
  
  Called at end of each HERMES run

- Recoil::End()
  - RecoilAnalyzer::End()
  
  Called once at end
Analysis Modules

- Can/Must implement
  - `Begin()`
    - Create histos etc.
    - Add branches to output tree
    - Create slave analysis modules
  - `BeginOfRun()`
  - `Process()`
    - Analysis of data is done here
  - `EndOfRun()`
  - `End()`
    - e.g. fit histos

- Registered with the analysis object
  - Processed for each event

- Slave module of another analysis module
  - Processed on demand
• The HERMES Recoil Detector
• General Idea
• A brief Project Overview
• Selected Features
  – uDST Framework
  – Cut Management
  – Event Display
• Summary and Future Plans
RecoilUtils Overview

- **HCommon**: Common Classes
- **HGeom**: HERMES Geometry Interface
- **XMas**: Extra Math
- **RootEx**: ROOT Extension
- **RDConfig**: Configuration Files
- **HAnalysis**: XTC Analysis
- **uDST**: uDST Analysis
- **ED3D**: Recoil Event Display
- **ELoss**: Energy Loss Calculations
- **HQt**: QT based GUIs
RecoilUtils Overview

**HCommon**
Common Classes

**HGeom**
HERMES Geometry Interface

**XMas**
Extra Math

**RootEx**
ROOT Extension

**RDConfig**
Configuration Files

**ED3D**
Recoil Event Display

**ELoss**
Energy Loss Calculations

**HAnalysis**
XTC Analysis

**uDST**
uDST Analysis

**HQt**
QT based GUIs

- 865 source files
- 85000 lines of code
- 319 classes
● The HERMES Recoil Detector
● General Idea
● A brief Project Overview
● Selected Features
  – uDST Framework
  – Cut Management
  – Event Display
● Summary and Future Plans
The uDST Framework

• Used for physics analysis

• Very similar to XTC framework
  – Main analysis object is called *uDST*
  – Analysis modules derived from *uDSTAnalyzer*

• But...
  – uDSTs contain only stripped information (no raw data and hits)
  – uDSTs contain slow control information
  – Data is split into bursts (10 s)
The uDST Framework

ADAMO to ROOT Interface

uDSTSelector

uDST Analysis Object

ROOT Files

ROOT Tree

uDSTwriter
The uDST Framework

- Additional methods in uDSTAnalyzer
  - `BeginOfBurst()`
  - `EndOfBurst()`

- Certain bursts may be skipped due to data quality

- Introduce burst selector base class `uDSTVBurstSelect`

- Burst selector must implement `IsGoodBurst()`
  - Burst selector is processed at beginning of each burst
  - Burst is skipped if `IsGoodBurst()` returns `false`

```c
void AnaInit()
{
  uDST::Get("uDSTDemo");

  guDST->RegisterBurstSelect("ClassName", "Name", "Option");

  guDST->RegisterAnalyzer("ClassName", "Name", "Option");
  ...
}
```
Outline

• The HERMES Recoil Detector
• General Idea
• A brief Project Overview
• Selected Features
  – uDST Framework
  – Cut Management
  – Event Display
• Summary and Future Plans
Cut Management

• Analysis modules can be used by different people
  – Different analyzers might want / try different cuts

• Cuts must not be hidden somewhere in the code
  – Documentation
  – Transparency

• Values of cuts should not be hard-coded

• Using a different set of cuts should work without recompiling the code
Cut Management

• All cuts declared in header files of analysis modules

• Substitute “basic” types by corresponding cut classes
  
  – $\text{Double}_t \rightarrow \text{THCutD}$
  
  – $\text{Int}_t \rightarrow \text{THCutI}$
  
  – $\text{TF1} \rightarrow \text{THCutF1}$

• Cut classes provide all methods and operators known from “basic” types

• Set default values for cuts in constructor of analysis module

• At startup: analysis modules register cuts with a cut manager (TCutManager)
TCutManager

- TCutManager stores cuts in folders
- Cuts are identified by
  - Name and type of the cut
  - Name of the analysis module
  - Classname of the analysis module
  - Path in analysis module hierarchy
- Provides XML IO
  - All cuts with value and description in one file
  - Cuts can be loaded at startup
- Allows multiple analysis modules of same type but with different cuts
Cut Management – An Example

DemoModule.h

class DemoModule : public RecoilAnalyzer
{
 public:
    DemoModule(const char * name, const char * option);
    virtual ~DemoModule();

    virtual void Process();
    ...

 protected:
    THCutD DemoDoubleCut; // Demo1
    THCutI DemoIntCut[2]; // Demo2

    ClassDef(DemoModule, 0)
};

DemoModule.C

DemoModule::DemoModule(const char * name, const char * option)
: RecoilAnalyzer(name, option)
{
    DemoDoubleCut = 3.75;
    DemoIntCut[0] = -12;
    DemoIntCut[1] = -13;
}

    ...

    void DemoModule::Process()
    {
        if (DemoDoubleCut>2.5)
            do something
                ...
    }

void AnaInit()
{
    Recoil::Get("DesyITSeminarDemo");

    gRecoil->RegisterAnalyzer("DemoModule", "SeminarDemo", "no option for now");
}
Cut Management – An Example XML File

- Analysis module header file used for documentation of cuts
- Dictionary provides all information during run-time
  - Type of cut
  - Name of cut
  - Comment Documentation
Cut Management

• All cuts are saved to output root file
  – With directory structure
  – Cuts are “browseable”
• The HERMES Recoil Detector
• General Idea
• A brief Project Overview
• Selected Features
  – uDST Framework
  – Cut Management
  – Event Display
• Summary and Future Plans
ED3D – The Recoil Event Display

- Uses ROOT GUI classes and TGeoManager
- Allows multiple independent 3D views
- Tooltip information for selected tracks and spacepoints
- Filter on track and event parameter
- Bookmarks
ED3D – Tooltips

- Tracks and spacepoints are “selectable”
- Tooltips show basic information
  - Tracks: Momentum, Angles and Vertex
  - Spacepoints: Energy and Position
• Event information is printed in an extra window
• For a “selected” track all spacepoints with energies and coordinates are shown
• For a “selected” spacepoint all associated tracks are shown
ED3D – Track Filter

- Display only tracks/events that fulfill certain conditions
- Extendable by *user filters*
  - Code will be compiled on startup of event display
  - Filters will appear in GUI
ED3D – OpenGL View

- Uses ROOT's standard OpenGL viewer
- Tracks and spacepoints are not selectable
• The HERMES Recoil Detector
• General Idea
• A brief Project Overview
• Selected Features
  – uDST Framework
  – Cut Management
  – Event Display
• Summary and Future Plans
Summary and Future Plans

• XTC Framework
  – well tested
  – heavily used

• uDST Framework
  – needs a bit more testing
  – first DVCS analysis is currently done

• Geant4
  – offers different low energy models (interesting for Silicon Detector)

• uDST Framework
  – need more tested analysis modules

Get more people to use the software