

Programming Reconstruction Software for Large Computing Projects Course

Course syllabus and coding project descriptions

Course Title:

Programming Reconstruction Software for Large Computing Projects

Instructor:

Matthew Herndon

Course web pages:

Redmine <https://cdcv.s.fnal.gov/redmine/projects/fnal-soft-school-summer-2014/wiki/Wiki>

Indico: <https://indico.fnal.gov/conferenceDisplay.py?ovw=True&confId=8568>

GitHub: <https://github.com/herndon/FNALComp/tree/production>

CodeBrowser: <http://lpc.fnal.gov/FNALsoftwareSchool/CodeBrowser/index.html>

Syllabus

Day 0

Reading

Reading on best practices, and coding conventions used in this course. Reading on toy detector geometry and data objects used in the course.

Exercise 0:

Introduction, Setup accounts, compile, run, modify course software

See Day 0 web page:

<https://cdcv.s.fnal.gov/redmine/projects/fnal-soft-school-summer-2014/wiki/Day0>

Day 1

Introduction

Introductory lecture on course goals, course structure, and day 1 course work

Exercise 0 review

Review the pre course exercise

Lecture

Lecture on course code infrastructure and coding in a large scale software project.

Lecture

Hit reconstruction and relevant software engineering topics such as data/algorithm abstraction.

Project 1 planning

Small group coding project cluster and hit finding.

See project descriptions below

Project planning jointly by coding groups and TA/Instructor

Project 1 coding

Group project coding

Project 1 assessment
Review of progress with TAs and Instructor

Reading
Review of software school code, code documentation and best practices

Day 2

Brief Introduction
On day 2 course work

Project 1 review
Review and discuss project 1

Lecture
Lecture on issues in hit finding and how hit properties effect track reconstruction. Data encapsulation, data object creation, data/algorithm separation

Project 2: Hit finding performance and effects on tracking
planning, coding, assessment
Group coding project on evaluating the performance of the hit finding and applying the results for tracking reconstruction.
See project descriptions below

Reading on tracking algorithms and track fits

Day 3

Brief Introduction
On day 3 course work

Lecture
Track reconstruction with focus on track candidate building. Discussion of types detectors and associated of track candidate finding algorithms. Includes relevant topics on planning integrated sets of reconstruction algorithms.

Project 3: Tracking candidate reconstruction
planning, coding, assessment
Group coding project on track candidate reconstruction. Students will program an algorithm to find track candidates using reconstructed hits.
See project descriptions below

Reading on iterative tracking algorithms and track fits

Day 4

Brief Introduction
On day 4 course work

Project 3 review
Review and discuss project 3

Lecture

Discussion of performance metrics in tracking. Assessment of both code execution speed and tracking performance. Track reconstruction with focus on iterative tracking. Relevant topics on execution speed and code maintainability

Project 4: Tracking performance

planning, coding, assessment

Group coding project on evaluating tracking performance speed, efficiency, resolution.

See project descriptions below

Day 5

Brief Introduction

On day 5 course work

Project 4 review

Review and discuss project 4

Lecture

Short lecture on data visualization. Relevant topics on integrating tools for debugging coding errors and performance issues.

Project 5: Event Display

planning, coding, assessment

Students will extend a module to perform event display.

See project description below.

Detailed project descriptions:

FNAL Software School coding projects

Day 0 pre course exercise: introduction

Short description: The exercise involves compiling and running an example programs that generates track and detector data and a second program that reads the generated data. The students should then modify a program Modules to histogram track, hit and strip information. The students must complete the exercise before the first day of class.

Input: StripSet, GenHitSet, and GenTrackSet which contain the generated track data and the simulated response of the detector to that data.

Output: Histograms of data object properties in three Sets.

Notes: The students will modify Day0HistogrammingModule to access the data objects and histogram a list of properties from those objects.

Purpose: The primary purpose of this exercise is to provide resources for the students to introduce themselves to the Fermilab computing environment, the course code infrastructure, C++ programming, and best practices in programming and apply those resources in a pre course exercise where they download and then compile, run, and modify the course software. Completing the exercise will establish that the student meets the course prerequisites.

For full instructions see:

Day0 web page: <https://cdcv.fnl.gov/redmine/projects/fnl-soft-school-summer-2014/wiki/Day0>

Day 1 course exercise: Cluster and hit finding applying the idea of algorithm abstraction.

Description: This exercise involves cluster finding followed by Hit finding using the strip data contained in the StripSet object.

a) Cluster finding

Cluster finding algorithm: The students will design an algorithm that identifies all sets of adjacent strips with an above threshold adc value on each layer of the strip detector. Each cluster is due to the charge deposition of one, or possible more than one charged track, when they traversed the sensor.

Input: The input of the exercise is the StripSet.hh, which consists of a vector of maps, one map per each layer, where the map contain pairs of strip number and adc counts.

Output: The output of the exercise could be a vectors of initialStrips and a vector of vectors where the vectors have acd values for the adjacent strips. One data object of this type should be created per layer in a loop over the detector layers.

b) Hit finding

Hit Finding: The students use the output of the previous step to create Hits and place them in a HitSet.

Input: the data created in the cluster finding step.

Output: A HitSet containing one Hit per cluster.

Notes: Free functions exist to calculate a cluster centroid in a strip number coordinates and also to convert from strip number coordinates to local distance coordinates and to global coordinates. The students will use these functions to calculate the global position coordinates of the cluster, create a Hit with position, layer, number of strips, a goodHit designation (set to true for now) and resolution information and store the result in the HitSet. The students should familiarize themselves with the data objects and associated function code for the StripSet, HitSet and Hits as well as the DetectorGeometry and associated free functions.

Purpose: The purpose of this exercise is to familiarize the students with object oriented algorithm programming in a practical exercise. Logical tasks in the cluster and hit finding algorithm should be outlined and functions written for each where the functions form a hierarchy of abstractions that map onto the ideas that are part of the cluster and hit finding problem being solved.

Day 2 course exercise, hit finding performance and tracking

Short Description: The purpose of this exercise to access the performance of the hit finding and use those results to fill in essential information for the tracking reconstruction code. Efficiency and resolution will be used as metrics of performance. The resolution information is essential to the tracking reconstruction, which requires accurately estimated resolutions to function correctly. Expected hit resolutions for different classes of Hits and the effects of hit efficiency and resolution in tracking will be discussed.

Coding: The students use and modify HitCompareModule to produce the full set of performance histograms.

a) Hit performance metrics:

Input: The generated and reconstructed Hit Sets are used as input to HitCompareModule.

Output: The output will be histograms of the efficiency of hit finding, histograms differences between the measured and true generated hit positions, and resolutions extracted from that information. Resolution histograms will be calculated separately for different “good” and “bad” classes of Hits.

b) Improving track reconstruction

Coding: The students will modify the `sensorgeometry.txt` to enter resolutions for poor resolution classes of Hits. Then the students will then modify the hit reconstruction code to assign these resolutions where appropriate.

Input: Hit resolutions from the previous step in this project.

Output: Improved resolution model in `sensorgeometry.txt` and improve estimated resolutions in Hit reconstruction.

Note: We will run the default track reconstruction before and after this project and use `TrackCompareModule` to access the effect of our work.

Purpose: A key issue illustrated by this exercise is planning necessary to integrate a set of many data objects, associated functions, and Modules to produce and consume data in order to meet primary project goals such as efficient and good with well understood resolution tracking.

Day 3 course exercise

Short Description: This exercise involves track candidate finding. Track candidate finding involves finding a minimum set of measurement points, Hits, which can be used to construct a helix, which is the trajectory of a charged particle in a magnetic field. Given measurements in X (x-y since y is give by the layer location) and Z (z-y), you need three X points to form a circle showing how the particle bends in the magnetic field due to it’s transverse momentum, and two Z points, to find the z momentum component, in order to define the helix.

a) Track candidate finding

Algorithm: The track candidate finding algorithm should read hits from the `HitSet` and use hits from the outer three layers of the detector and find all combinations of 3 X and 2 Z hits. The existing algorithm, which uses two X coordinate hits and 1 hit with Z information and the primary vertex X and Z information, to form a track candidate can be used as an example. The algorithm could be expanded to more layers to account for detector inefficiency.

Input: The reconstructed `HitSet` containing the Hits on all layers.

Output: a vector of `vector<int>` one for each candidate where each vector has of 5 Hit indices.

b) Track candidate set construction

Coding: The students will take the output of the first step and convert it into a set of candidate Tracks using the existing track candidate algorithm as an example

Input: the vector of vectors from the previous step in the exercise.

Output: A `TrackSet` which contains each of the candidate Tracks.

Notes: The output of the first step can be passed existing code that i) takes the fits Hits and builds a Track and ii) runs a selection on that track to access whether it is likely to be a real track.

Beyond the scope of this exercise candidate finding is followed by iterative reconstruction, which iterates through the remaining layers searching for compatible hits.

Purpose: The purpose of this exercise is to have the students engage in object oriented algorithm programing within and more complex structure of existing classes. Any candidate finding exercise involves a common set of tasks including identifying the Hits used to form the candidate, creating a Track helix from the Hits, and evaluating

whether the candidate is viable based selection criteria. These tasks have been abstracted as functions available in the Tracking code. The students will program an example of the critical function in the candidate finding process.

Day 4 tracking performance evaluation

Short Description: This exercise involves using infrastructure tools to evaluate the performance of candidate finding and iterative reconstruction algorithms. The students will evaluate the performance of these algorithms using speed and efficiency, both track hit finding efficiency and full track finding efficiency, as metrics. A profiler will be used to study execution speed. The TrackCompare module will be used and possibly expanded to perform efficiency studies. From the existing infrastructure along with the results of the Day 3 exercise different candidate finding algorithms and track reconstruction choices can be compared.

Input:

The input will be TrackSets of track candidates produced by the existing or the new student designed candidate finding algorithms. These inputs will be used in TrackCompareModule to access track candidate finding performance in terms of efficiency and resolution by comparing to GenTracks. Also the candidate TrackSets will be run through tracking reconstruction, which performs an iterative search over the layers, to form TrackSets of full Tracks with Hits on all layers. These reconstructed tracks in a TrackSet will also be used as input to TrackCompareModule. A profiler will also be run to measurement the execution speed of track candidate finding and track reconstruction.

Output: The output will be efficiencies as measured by TrackCompareModule, histograms accessing track properties such as resolutions, and measurements of execution speed.

Notes: These students will have to modify the main programs to run the desired algorithms through instances of TrackCompareModule and, if necessary, extend TrackCompareModule to produce more output.

Purpose: As with Day 2 this exercise uses a practical example to pursue the concept of project planning to meet primary project goals.

Day 5 event display

Short Description: This exercise will involve using root interfaces to display data. Hit and Track objects will already have interfaces for root display(in progress) we will extend this to include GenTracks and then use this as a tool to study specific instances of Tracking failures in terms of inefficiencies to find complete tracks, track hits, or poor resolution.

Input: GenTrackSet produced by the dataGen program

Output: Root event display include GenTracks

Purpose: The focus of this exercise will be an example of anticipating problems in software or tracking reconstruction and designing or utilizing existing tools to surmount them