Development of Machine Learning Algorithms and Software Implementations for Reconstructing Straight Line Trajectories

The principal goal of this research project will be to develop machine-learning algorithms and software implementations that can replace some of the most computationally expensive parts of event pattern recognition (charged track reconstruction). LHCb reconstruction algorithms take low-level data from the detector and produce high-level objects corresponding to physical particles that traverse the detector. For example, charged-particle reconstruction determines the momentum (magnitude and direction) of each such particle along its trajectory; its 3-D origin and direction; and the 3-D location of each proton-proton collision (the primary vertices). This is the most important reconstruction algorithm in the trigger in terms of physics and CPU time. It runs over all events and consumes most of the first stage trigger CPU budget.

For Run 3 of the LHC, scheduled to start in 2021, the first charged tracking detector element in LHCb, the VELO, will be upgraded to use pixels rather than strips. The detector sits in a region of space with neglible magnetic fields, so the trajectories of the charged particles are essentially straight lines. The four key goals of the VELO reconstruction are

- associating detector hits with tracks;
- determining the slopes and intercepts of the individual tracks, along with their corresponding error matrices;
- associating which tracks emerge from which primary interaction vertices (or none);
- determining the positions of the primary vertices, along with their corresponding error matrices.

Although these goals are listed separately, they are interdependent in many ways. Exisiting algorithms find tracks with very high efficiency and fidelity, but they do not execute quickly enough to allow all tracks from all events to be reconstructed given the experiment's limited budget for computing hardware. Several types of machine learning algorithms have been proposed as alternative approaches for achieving the same goals with significantly faster inference time. Recurrent neural networks (RNNs) should be able to determine the slopes and intercepts of individual tracks, along with their error matrices, for a specifed set of hits. The focus of this project will be producing the data sets needed to properly study this problem, documenting them, putting them on github so others can work on it too, and providing at least one working implementation using an RNN. This code will also be carefully documented and shared on github. If there is sufficient time, the student will try a variety of RNN implementations, explore their performances in terms of physics fidelity and in terms of resource consumption.

The principal mentors for this project will be Prof. Mike Williams from MIT and his Ph.D. student Constantin Weisser. In addition, the student will collaborate with Dr. Vava Gligorov from the University of Paris and Dr. Henry Schreiner from the University of Cincinnati. The work will be done while in residence in Cambridge, MA.

Deliverables and Milestones

This project will extend over 12 weeks. Before the project begins, the mentors will prepare a data set of simulated tracks to be studied. The primary deliverable will be an RNN-trained inference engine that mimics the fidelity of the traditional Kalman filter code. The secondary (but absolutely critical) devliverable will be the documentation.

Week 1	set up and run traditional Kalman filter on test data set and determine benchmark targets;
Weeks 2-3	run existing "deep Kalman" RNN implementation on a simulated data set; compare to track parameter and timing benchmarks;
Weeks 4-6	develop purpose-built RNN for tracking;
Weeks 7-9	work on strategies for speeding up execution (e.g., identify fake track candidates early);
Weeks 10-11	work on speeding up execution at the software level (profile performance, change activation function, etc.);
Week 12	document test data sample, benchmarks, and the final RNN implementation on github so others can use and/or extend it.