Track fitting/finding with \textit{acts}

Xiaocong Ai, Heather Gray for the ACTS developers
UC Berkeley

IRIS-HEP topical meeting, Feb 19, 2020
The tracking challenge

- Much increased combinatorics at future hadron colliders
  - Great strain on CPU
  - More complex tracking environment
- All possible efforts to reduce memory footprint and speed up track reconstruction
  - Thread safety for concurrency
  - Performant software technology (data storage memory access)
  - Fast tracking strategy (ML techniques)
  - Benefit from multi-core and accelerators

<μ> up to 200 (~5k particles/event)

More sensitive to rare physics, and far more combinatorics!
ACTS recap

- A tracking toolkit for future detectors and modern computing infrastructure
  - Based on ATLAS tracking code (well-tested, but difficult maintenance, thread-unsafe design etc.)
  - Improved performance
    - Modern C++ 17 concepts
    - Thread-safe design
    - Minimal dependencies for ease of integration
    - Ease of use and maintenance

- A testbed for new algorithms (ML techniques) and hardware architectures

Tracking ML detector with ACTS fast simulation

A Common Tracking Software (Acts)

http://acts.web.cern.ch/ACTS/

Latest release: v0.16.00
05 Feb 2020
ACTS components and functionalities

- Geometry
  - Tracking geometry following ATLAS concepts
  - Plugins: DD4hep, Tgeo, GeoModel

- Event Data Model
  - Eigen-based tracking EDM

- Propagator
  - Integrate transport and navigation
  - Highly templated to avoid deep look-up

- Tracking features
  - Track seeding and track following (CombinatorialKalmanFilter)
  - Track fitting (KalmanFilter, Gaussian Sum Filter)
  - Vertex Finding&Fitting

Focus of this talk
TrackState EDM in "ats"

- Design supports concept of KalmanFilter
- A container of possible information:
  - track parameters (predicted, filtered, smoothed)
  - measurements (uncalibrated, calibrated)
  - fitting quality
  - path Length
  - typeFlags (Parameter, Measurement, Outlier, Hole, Material etc.)
Track EDM in \textit{acts}

- Eigen::Array based container (Acts::MultiTrajectory) of track states for efficient storage/look-up
  - Keep track of capacity and size, provide read-write views into storage with index-tuples
- Allows branching of tracks. Facilitate extension to Combinatorial Kalman Filter
- Avoid storage duplication for shared measurements and parameters

```cpp
// single items
using Coefficients = Eigen::Matrix<Scalar, Size, 1, Flags>;
using Covariance = Eigen::Matrix<Scalar, Size, Size, Flags>;
using CoefficientsMap = Eigen::Map<ConstIf<Coefficients, ReadOnlyMaps>>;
using CovarianceMap = Eigen::Map<ConstIf<Covariance, ReadOnlyMaps>>;
// storage of multiple items in flat arrays
using StorageCoefficients =
  GrowableColumns<Eigen::Array<Scalar, Size, Eigen::Dynamic, Flags>, SizeIncrement>;
using StorageCovariance =
  GrowableColumns<Eigen::Array<Scalar, Size * Size, Eigen::Dynamic, Flags>,
                  SizeIncrement>;

struct IndexData {
  using IndexType = uint16_t;

  static constexpr IndexType kInvalid = UINT16_MAX;

  IndexType irefsurface = kInvalid;
  IndexType ipprevious = kInvalid;
  IndexType ippredicted = kInvalid;
  IndexType ifiltered = kInvalid;
  IndexType ismoothered = kInvalid;
  IndexType ijacobians = kInvalid;
  IndexType ipjetector = kInvalid;

  double chi2;
  double pathLength;
  TrackStateType typeFlags;

  IndexType iuncalibrated = kInvalid;
  IndexType icalibrated = kInvalid;
  IndexType icalibratedsourceLink = kInvalid;
  IndexType measdim = 0;
};
```
KalmanFilter in *ats* (Cont’d)

- Used as an *Actor* in propagator
- Templated on propagator, KalmanUpdater, KalmanSmoother, outlier finder, measurement calibrator etc.
- Support hole search and outlier rejection during the fitting

```cpp
template <typename propagator_t, typename updater_t = VoidKalmanUpdater, 
typename smoother_t = VoidKalmanSmoother, 
typename outlier_finder_t = VoidOutlierFinder, 
typename calibrator_t = VoidMeasurementCalibrator, 
typename input_converter_t = VoidKalmanComponents, 
typename output_converter_t = VoidKalmanComponents>
class KalmanFitter {
```
KalmanFilter in acts

- One fit invocation takes:
  - Starting parameter (with large uncertainty)
  - Measurements (unsorted source links, e.g. clusters) on a known trajectory
  - User-defined fitting options
    - Algorithm context (GeometryContext, MagneticFieldContext, CalibrationContext)
    - Target surface
    - Outlier search criteria
    - Material effects
    - Smoothing mode

- Fitting result:
  - An Acts::MultiTrajectory object with single track entry point
  - Fitted parameter

```cpp
KalmanFilterOptions(std::reference_wrapper<GeometryContext> gctx,
                     std::reference_wrapper<MagneticFieldContext> mctx,
                     std::reference_wrapper<CalibrationContext> cctx,
                     const OutlierFinderConfig& ofCfg,
                     const Surface* rSurface = nullptr,
                     bool mScattering = true, bool eLoss = true,
                     bool bwdFiltering = false)
```

```cpp
geoContext(gctx),
magFieldContext(mctx),
calibrationContext(cctx),
outlierFinderConfig(ofCfg),
referenceSurface(rSurface),
multipleScattering(mScattering),
energyLoss(eLoss),
backwardFiltering(bwdFiltering) {}
KF fitting efficiency

- Fitting efficiency is defined as: \( \frac{N_{reco}}{N_{truth}} \)
- ~100% fitting efficiency with \(|\eta|<1.5\)
KF fitting resolution

- Single muon, $0.1<p_T<100$ GeV, $|\eta|<2.4$
- TrackML detector, ATLAS B field

Little material for TrackML detector, hence no great justification for gaussian pulls
KF timing test

- Running with two different approach for smoothing
  - So-called 'smoothing-matrix' formalism
    - Faster, relies on surface-wise Jacobian in forward filtering
  - Run another Kalman filter in backward direction
    - Slower, but more stable

- Single muon, 0.1<p_T<100 GeV, |η|<2.4
- TrackML detector, ATLAS B field
CombinatorialKalmanFilter in \( \text{acts} \) (Cont’d)

- KalmanFilter-extended track following algorithm
  - Simultaneous tracking fitting and finding
  - Allow track branching if more than one compatible measurement found
    - Support user-defined measurement search and branching strategy (e.g. detector-specific chi2 cutoff and maximum branches)
  - Allow stopping of bad quality branch

```cpp
/// @brief CombinatorialKalmanFilter implementation of Acts as a plugin
///
/// to the Propgator
///
/// @tparam propagator_t Type of the propagation class
/// @tparam updater_t Type of the kalman updater class
/// @tparam smoother_t Type of the kalman smoother class
/// @tparam source_link_selector_t Type of the source link selector class
/// @tparam branch_stopper_t Type of the branch stopper class
/// @tparam calibrator_t Type of the calibrator class
/// @tparam input_converter_t Type of the input converter class
/// @tparam output_converter_t Type of the output converter class

template<typename propagator_t, typename updater_t = VoidKalmanUpdater,
          typename smoother_t = VoidKalmanSmoother,
          typename source_link_selector_t = VoidSourceLinkSelector,
          typename branch_stopper_t = VoidBranchStopper,
          typename calibrator_t = VoidMeasurementCalibrator,
          typename input_converter_t = VoidKalmanComponents,
          typename output_converter_t = VoidKalmanComponents>

class CombinatorialKalmanFilter {
```
CombinatorialKalmanFilter in

• One track finding invocation takes:
  - Starting parameter (currently using smeared truth parameter, will be taken from seeding)
  - All measurements in a single event
  - User-defined track finding options
    • Algorithm context
    • Target surface
    • Source link selection criteria
    • Material effects
    • Option for smoothing

• Track finding result:
  - An Acts::MultiTrajectory object with a list of track entry points for all found tracks
  - (Optional) fitted parameter

  • Found tracks for ttbar at $\mu = 200$ (~6k truth particles, ~70k truth hits)
  • TrackML detector, ATLAS B field
CKF efficiency & fake rate

- Measurement selection based on $\chi^2$ window:
  - Sequential: branching with only the best measurement
  - Combinatorial: branching with all compatible measurements
- Track selection: $n_{\text{Hits}} > 5$, $|\eta| < 2.4$
- Reco-truth matching criteria:
  \[ \frac{N_{\text{hits}}(\text{Majority})}{N_{\text{hits}}(\text{Total})} > 0.5 \]
- Track finding efficiency is defined as:
  \[ \frac{N_{\text{reco}}(\text{selected, matched})}{N_{\text{truth}}(\text{selected})} \]
- Fake rate is defined as:
  \[ \frac{N_{\text{reco}}(\text{selected, unmatched})}{N_{\text{reco}}(\text{selected})} \]

TrackML detector, ATLAS B field
CKF timing test

- Timing varies with measurement selection strategy (which requires detector-specific tuning)
- Planned tests with more realistic detectors (eg. OpenData, ATLAS ITk)
- Investigate possible speed-up with fast tracking geometry

TrackML detector, ATLAS B fileld

$t\bar{t}$, $p_T > 1$ GeV, $\sqrt{s} = 14$ TeV
ACTS uses an AlgorithmContext object to support on-the-fly event-dependent changes of alignment, calibration and magnetic field.

```cpp
size_t algorithmNumber; // Unique algorithm identifier
size_t eventNumber; // Unique event identifier
WhiteBoard& eventStore; // Per-event data store

Acts::GeometryContext geoContext; // Per-event geometry context
Acts::MagneticFieldContext magFieldContext; // Per-event magnetic field context
Acts::CalibrationContext calibContext; // Per-event calibration context
```

Validated concept of contextual alignment and calibration.

Propagation tests with contextual **alignment**

(Different alignment every single event, n_threads = 4)

<table>
<thead>
<tr>
<th>Run the event loop</th>
<th>Start event 0</th>
<th>Start event 1</th>
<th>Start event 2</th>
<th>Start event 3</th>
<th>Start event 4</th>
<th>Start event 5</th>
<th>Start event 6</th>
<th>Start event 7</th>
<th>Start event 8</th>
<th>Start event 9</th>
</tr>
</thead>
</table>

Track fitting test with contextual **calibration**

(Different calibration every 10 events, n_threads = 8)

![Efficiency plot](image)

- single muon, |η|<2.4, without calibration
- single muon, |η|<2.4, with calibration

12 seconds 5 seconds
Much more progress in

- Close-to complete Vertexing Finding & Finding (B. Schlag)
  - Currently performance validated against Athena/Tracking
- Prototype of Gaussian Sum Filter as non-gaussian extension of KF (J. Zhang)
- Improvement of Fatras simulation (M. Kiehn, F. Klimpel)
  - Energy loss & multiple scattering validated
  - Hadronic interaction currently re-parameterised
  - Foreseen use of Geant4 for particle decay
- Optimization of repository infrastructure (A. Salzburg, M. Kiehn, P. Gessinger-Befurt, F. Klimpel et al.)
  - Improved EDM, geometry, stepper etc.
- Exploration of parallelism with multi-core and accelerator
  - Intra-event parallelism (G. Mania, N. Styles)
  - Seed finder with accelerator (V. Pascuzzi, C. Legett et al.)
Detector geometry with

- Detector implemented:
  - ATLAS ID+Calo, ATLAS ITK, sPhenix silicon, OpenData, TrackML. FCC-hh

- On-going/planned implementation:
  - ATLAS Muon System
  - Belle-II silicon (+ Drift Chamber)
  - sPhenix TPC
Summary

- Large increase in track multiplicity at future colliders needs high performance tracking software
- ACTS provides a framework-independent and detector-independent tracking toolkit with inherent thread-safety
  - Well-validated tracking infrastructure (geometry, propagator, EDM)
  - Expanding tracking toolkit
    - Pattern recognition, fitting, vertexing etc.
  - New experiment detector implementation with world-wide collaboration
- Work In Progress with ACTS:
  - Extended geometry concept for various HEP detector
  - Validation of tracking tools with different detector
  - Exploration of innovative tracking techniques and speed-up with hardware architectures
backup
**Extrapolator to Propagator**

- ATLAS Tracking SW:
  - Distinction between propagator (transport) and extrapolator (transport, navigation & material effects)

- Integrate transport and navigation into **propagator** (template design to avoid deep look-up)
  - **Stepper** integrates equation of motion and provides step estimation
    - **EigenStepper** (ATLAS adaptive Runge-Kutta stepper rewritten using Eigen) as the primary integrator
    - Extension for transport in dense volumes, e.g. calorimeter
  - **Navigator** determines navigation step size to reach detector
  - **Options** allows for custom implementation at integration step
    - Actors: e.g. **MaterialInteractor**, **KalmanActor**
    - Aborters: abort conditions