Studies on combined GNN + CKF tracking

12.10.2023

Benjamin Huth\textsuperscript{1}

Lukas Heinrich\textsuperscript{2}, Andreas Salzburger\textsuperscript{3}, Tilo Wettig\textsuperscript{1}

(1) Universität Regensburg, (2) TU München, (3) CERN
1) Motivation and idea
2) ACTS, the ODD & the Exa.TrkX pipeline
3) Experiment setup
4) Performance analysis
Motivation

- Graph Neural Network (GNN) based tracking
  - Very promising results achieved recently
  - But: issues observed with low resolution spacepoints [1], [2]

- How to solve this?
  - Idea: Use GNN only in pixels, follow up with Combinatorial Kalman Filter (CKF) → presented here
  - Heterogenous networks [2]

[1] 2022, B. Huth, Applying and optimizing the Exa.TrkX Pipeline on the OpenDataDetector with ACTS
[2] 2022, D. Murnane, Heterogeneous GNN for tracking
Idea

- **GNN:**
  - Resolve combinatorics with high resolution spacepoints in pixels
  - Use ordinary KF here

- **CKF**
  - Completes tracks in strips

- **Benefits of combination:**
  - High quality seeds without duplicates for CKF
  - Use CKF in region with lower density (→ less branching)
  - CKF can e.g. use single strip measurements
  - Smaller graph (pixel only)
ACTS & OpenDataDetector

- **ACTS** [1]:
  - Charged particle tracking software package
  - Usage: both production & R&D
  - This work based on the **ACTS examples framework**

- **OpenDataDetector (ODD)** [2]
  - Virtual silicon detector based on DD4hep
  - Structure:
    - Pixels: 2D, 15µm resolution
    - Short strips: 2D, 43µm/1.2mm
    - Long strips: 1D (stereo angle), 72µm

[1] github.com/acts-project/acts
GNN pipeline

- Training with custom branch of GNN4Itk common framework [2]
- Acts Exa.TrkX-plugin based on torchscript and Boost.Graph [3]

Graph construction
- metric learning approach

Edge classification
- simple MLP filter
- GNN

Track building
- connected components

---

[1] 2023, P. Calafiura, The Exa.TrkX project
[2] gitlab.cern.ch/bhuth/commonframework
KF-CKF-Hybrid implementation

- **Very easy** implementation (~200 lines C++)
  - custom *measurement-accessor* for CKF*
    - Use measurements from GNN prototracks in pixel volumes
    - Use all available measurements in strip volumes
  - Shows flexibility of ACTS algorithms

*actually: SourceLinkAccessorDelegate*
• Consider 4 different chains with different degrees of truth information:

- **CKF chain**
  - Standard Seeding
  - CKF
  - Prototrack to Seeds
  - Combination KF + CKF
  - KF
  - no truth info

- **GNN+CKF chain**
  - GNN tracking PIXELS
  - Prototrack to Seeds
  - Combination KF + CKF
  - no truth info

- **Proof of concept chain**
  - Truth tracking
  - Prototrack to Seeds
  - Combination KF + CKF

- **Truth tracking chain**
  - Truth tracking
  - Whole Detector
  - KF

**Setup**

- **Pythia8: ttbar, PU 200**
- **Geant4 Full Sim in ODD**
- **Digitization**
  - geometric in pixel (no cluster merging), smeared in strips

**Connecting The Dots 2023**
Pipeline training

- Target particles (up-weighted in training):
  - min pT: 1GeV
  - min pixel hits: 3
- Training:
  - 2K events (1500, 250, 250)
  - Features:
    - r, phi, z
    - cluster features (metric learning only)
  - Goal:
    - high target efficiency
    - high total purity
- 2nd GNN stage improved performance
Pipeline training

- Use low edge cut in early filter stages to preserve efficiency

- Hypothesis:
  - GNN2 gives improvement because graph is better balanced?
Connecting The Dots 2023

**Track finding performance**

- Plots based on the ACTS CKF Performance Writer
  - 20 events
  - No ambiguity solution

- Truth matching:
  - Particles with pT > 1GeV, #hits >= 7
    - Also apply track selection based on fitted momentum and found hits
  - Track is matched if > 50% of its hits belong to one particle
  - Otherwise: fake

- Efficiency not optimal overall
  - Loose particles both in GNN and CKF stage
  - CKF performance not yet understood fully (see also Andi’s Talk)
  - Drop at η=0 needs to be investigated

- GNN+CKF chain:
  - Almost zero fakerate
  - Still duplication from CKF stage
    - in proof-of-concept as well
Remove the C

- Remove combinatorial aspect
  - Minimal efficiency cost
  - Zero duplication rate
- **Again:** I did not try to optimize this for standard CKF

```python
acts.MeasurementSelector.Config(
    [(acts.GeometryIdentifier(), [], [chi2Cut], [nMeasurementsCut]))
```

10 (default) → 1

![Track Efficiency vs.Eta](image1.png)

![Duplication Rate vs.Eta](image2.png)
Caveat:
- CKF not optimized for this study (seed filtering, ...)
- CPU timing seems to be very machine dependent
- Large combinatoric overhead of vanilla CKF configuration
- GPU: Nvidia A100 40GB
### Combinatorics

<table>
<thead>
<tr>
<th>Stage</th>
<th>Seeds</th>
<th>Seed Pur</th>
<th>Seed Eff</th>
<th>Tracks</th>
<th>Tracks/Seed</th>
<th>Sel Tracks</th>
</tr>
</thead>
<tbody>
<tr>
<td>C(KF) stage</td>
<td>~100K</td>
<td>0.15</td>
<td>0.87</td>
<td>~490K</td>
<td>3.82</td>
<td>~10K</td>
</tr>
<tr>
<td></td>
<td>~1600</td>
<td>0.96</td>
<td>0.93</td>
<td>~1600</td>
<td>1.00</td>
<td>~850</td>
</tr>
<tr>
<td></td>
<td>~700</td>
<td>1.00</td>
<td>0.99</td>
<td>~700</td>
<td>1.00</td>
<td>~650</td>
</tr>
</tbody>
</table>

**Track selection:**
- min \( p_T \): 1 GeV
- min hits: 7

**CKF**

- \( \chi^2 \) cut: 15.0
- meas cut: 10

**GNN+CKF**

- \( \chi^2 \) cut: 15.0
- meas cut: 1

**Proof of concept**
Conclusion

- Combination of GNN + CKF easy to implement in ACTS
  - Example for flexibility of ACTS tools (+ examples framework)

- Tracking performance:
  - Very low duplication and fakerates
  - Suboptimal efficiency (both due to GNN and CKF)

- Promising compute performance
  - Even more interesting, if CKF on GPU is available

Special thanks to Lukas Heinrich and MPG for providing access to GPU clusters

Reproducible workflow via snakemake
ACTS Exa.TrkX plugin

- Supports CPU & GPU
  - torchscript / FRNN / Boost.Graph
- Examples-framework integration
  - Supports cluster features
  - Edge-metrics hook based on truth sim data

```
20:25:22 MetricsHook INFO Metrics for total graph:
20:25:22 TrackFinding INFO Efficiency=0.195985, purity=0.983137
20:25:22 MetricsHook INFO Metrics for target graph (pT > 1 GeV, nHits >= 3):
20:25:22 TrackFinding INFO Efficiency=0.972825, purity=0.405324
```
Geometric digitization:

- Compute paths through pixels
- Path corresponds to charge deposit
- Configurable parameters:
  - Path threshold:
  - path smearing: $N(0,$

4 cluster features for Metric learning:

- Cell-count
- accumulated cell-activation
- Cluster size in $l_0$ and $l_1$
Improvements with cluster features

- With cluster features
  - Comparable efficiency
  - Improved purity
Different models / configurations

<table>
<thead>
<tr>
<th>Label in plot</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>„high eff“</td>
<td>setup presented before</td>
</tr>
<tr>
<td>„125 thickness“</td>
<td>Setup with 1 GNN</td>
</tr>
<tr>
<td>„no threshold“</td>
<td>Setup with 1 GNN + idealized geometric digitization (no threshold, no charge smearing)</td>
</tr>
</tbody>
</table>
Details on implementation

- CKF can be configured by SourceLinkAccessor template
  - Returns source-links / measurements for surface

```cpp
struct ProtoTrackSourceLinkAccessor
  : GeometryIdMultisetAccessor<IndexSourceLink> {
  Container protoTrackSourceLinks;

  auto range(const Acts::Surface& surface) const {
    const auto& logger = *loggerPtr;

    if (protoTrackSourceLinks.contains(surface.geometryId())) {
      auto [begin, end] = 
        protoTrackSourceLinks.equal_range(surface.geometryId());
      return {Iterator{begin}, Iterator{end}};
    }
    auto [begin, end] = container->equal_range(surface.geometryId());
    return {Iterator{begin}, Iterator{end}};
  }
};
```
Where do we lose tracks?

- Disentangle unphysical track candidates
  - Some graph algorithms tested in python but not yet implemented in ACTS
- CKF performance not yet 100% understood