Seeding & Pattern - ACTS Seed Finder

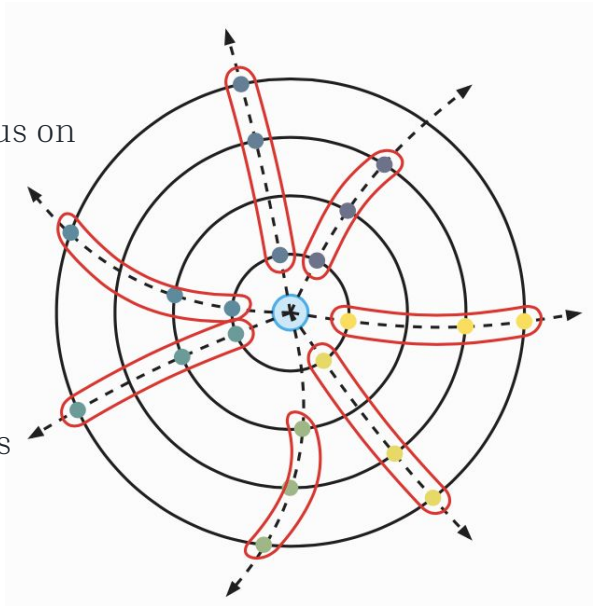
Luis Falda Coelho

ACTS Workshop

Sep 26th, 2022

The Seeding Algorithm in ACTS

Seeding implementation in ACTS:



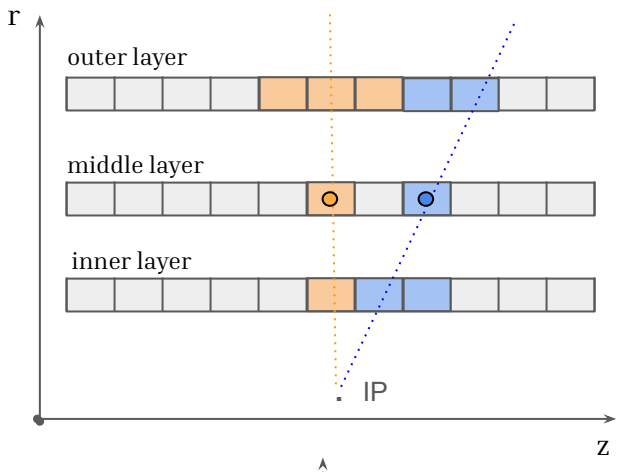
- *Core/include/Acts/Seeding/* based on the ATLAS track seeding with a focus on parallelism, maintainability and as detector agnostic as possible
- Optimised for ITk upgrade
- Can be extended for other experiments
- Track seeds are created by **combining three SPs**
- The seeds define the helical path of a charged particle in a homogeneous magnetic field
- Used by the tracking to search for additional measurements to create a track
- Various constraints and confirmation conditions are applied to reduce the number of seed candidates → decreasing the tracking time
- Cuts are configurable

SP Grid and Group Formation

The SPs in each detector layer are projected on a **rectangular grid (φ, z) of configurable granularity** (SpacePointGrid.ipp)

Triplet formation and search for seeds:

- Starts from **selecting SP in the middle detector layer. Then matching SPs are searched in the inner an outer layers**
- Grouping of the SPs in the grid allows to limit the search to neighbouring grid cells improving significantly algorithm performance
- The number of neighboring bins used in the search can be defined for each (φ, z) bin of the grid:



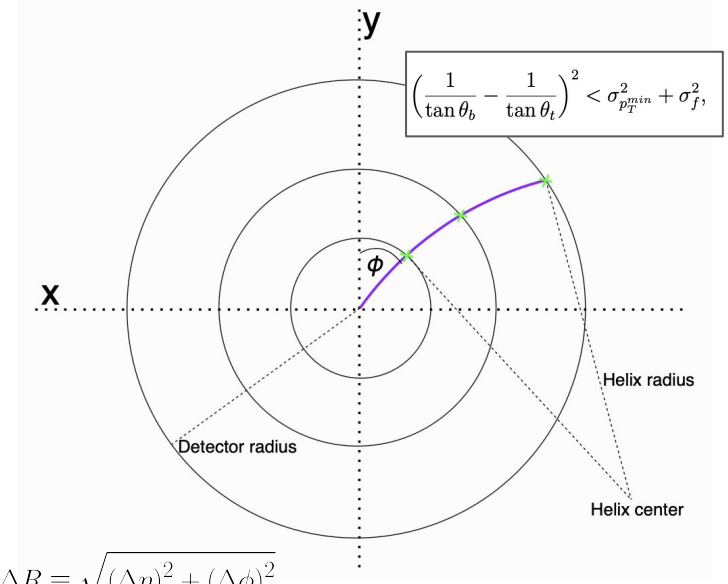
Configure the search based on geometrical assumptions

```
zBinEdges = [-3000.0, -2500.0, -1400.0, -925.0, -450.0, -250.0, 250.0, 450.0, 925.0, 1400.0, 2500.0, 3000.0,] # zBinEdges enables
    non-equidistant binning in z, in case the binning is not defined the edges are evaluated automatically using equidistant binning
zBinNeighborsTop = [[0, 0],[-1, 0],[-1, 0],[-1, 0],[-1, 0],[-1, 1],[0, 1],[0, 1],[0, 1],[0, 1],[0, 0],]  # allows to specify the number of
    neighbors desired for each bin, [-1,1] means one neighbor on the left and one on the right, if the vector is empty the algorithm returns
    the 8 surrounding bins
zBinNeighborsBottom = [[0, 1],[0, 1],[0, 1],[0, 1],[0, 1],[0, 0],[-1, 0],[-1, 0],[-1, 0],[-1, 0],[-1, 0],]
numPhiNeighbors = 1 # number of neighboring bins in phi direction
```

Seed Finder

Seed finder (Seedfinder.ipp) receives three iterators constructed from SPs from middle, inner and outer layers:

- *createSeedsForGroup* function starts by **iterating over SPs in the middle layer**
- **Configurable cuts are applied** to individual SPs and to combination of SPs:
  - Cuts to middle SPs if outside the region of interest
  - SP duplets are tested for compatibility by applying cuts with two SPs only (η, z_0, ΔR between SPs, compatibility with IP)



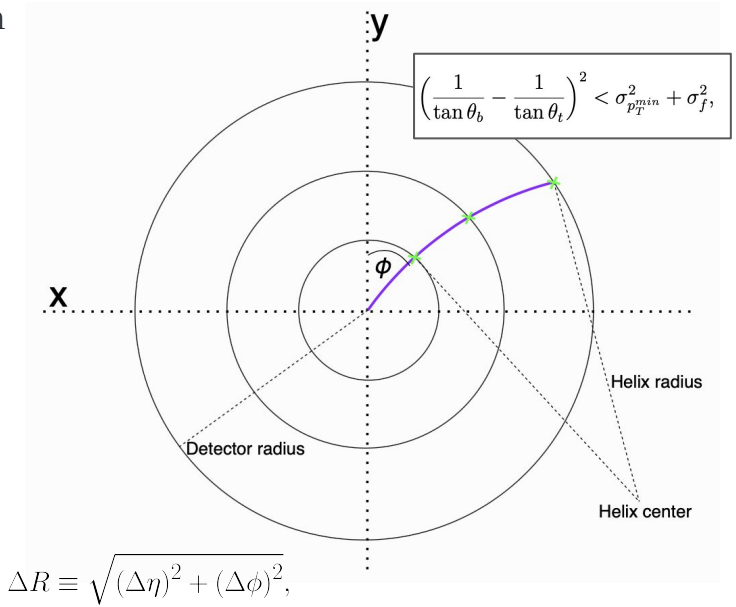

```
for (auto topSP : topSPs) {
  float rT = topSP->radius();
  float deltaR = rT - rM;
  // if r-distance is too small, try next SP in bin
  if (deltaR < m_config.deltaRMinTopSP) {
    continue;
  }
```

```
// check if duplet origin on z axis within collision region
float zOrigin = zM - rM * cotTheta;
if (zOrigin < m_config.collisionRegionMin ||
    zOrigin > m_config.collisionRegionMax) {
  continue;
}
```

Seed Finder

Seed finder (Seedfinder.ipp) receives three iterators constructed from SPs from middle, inner and outer layers:

- *createSeedsForGroup* function starts by **iterating over SPs in the middle layer**
- **Configurable cuts are applied** to individual SPs and to combination of SPs:
  - Cuts to middle SPs if outside the region of interest
  - SP duplets are tested for compatibility by applying cuts with two SPs only (η, z_0, ΔR between SPs, compatibility with IP)
  - Compatibility cuts between SP-triplets (ex: curvature compatibility with minimum p_T scattering, transverse impact parameter d_0)
- **Reduce the number of potential seeds** that may not lead to high quality tracks

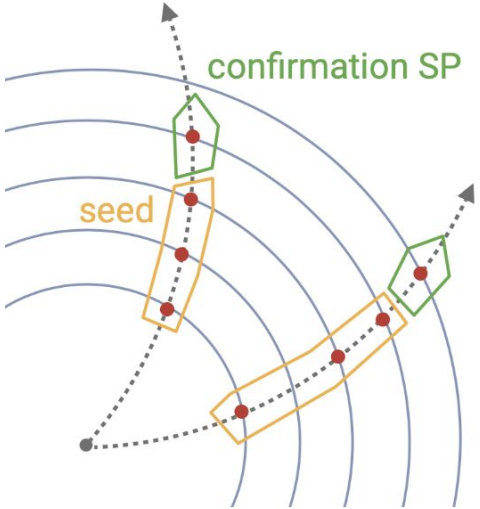


Seed Filter and Seed Confirmation

After selecting the SP-triplets, a seed filter/confirmation procedure is applied to all the triplet combinations to **rank the seeds based on a weight** and keep only the best seeds:

$$w = -c_1 \cdot d_0 + (c_2 \cdot N_t - |z_0|)$$

Seeds can also be classified as higher quality seeds if they have specific values of d_0, z_0 and N_t inside a configurable range of parameters that also depends on the region of the detector (i.e. forward or central region)



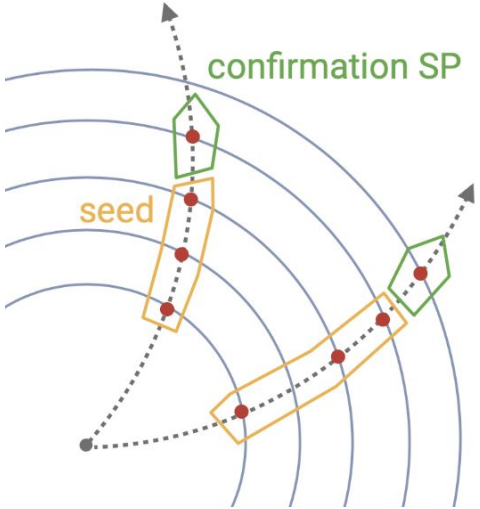

```
struct SeedConfirmationRangeConfig {
  // z minimum and maximum of middle component of the seed used to define the
  // region of the detector for seed confirmation
  float zMinSeedConf =
      std::numeric_limits<float>::min();  // Acts::UnitConstants::mm
  float zMaxSeedConf =
      std::numeric_limits<float>::max();  // Acts::UnitConstants::mm
  // radius of bottom component of seed that is used to define the number of
  // compatible top required
  float rMaxSeedConf =
      std::numeric_limits<float>::max();  // Acts::UnitConstants::mm
  // number of compatible top SPs of seed if bottom radius is larger than
  // rMaxSeedConf
  size_t nTopForLargeR = 0;
  // number of compatible top SPs of seed if bottom radius is smaller than
  // rMaxSeedConf
  size_t nTopForSmallR = 0;
```

Seed Filter and Seed Confirmation

A limit to the number of seeds produced for each middle SP is also applied

```
int maxQualitySeedsPerSpMConf = std::numeric_limits<int>::max();
```

If this limit is extrapolated, the algorithm will check if there is a seed with smaller weight that can be removed



Every SP holds the weight of the best seed containing that SP → one seed is kept only if its weight is greater than the weight of at least one of its SP components:

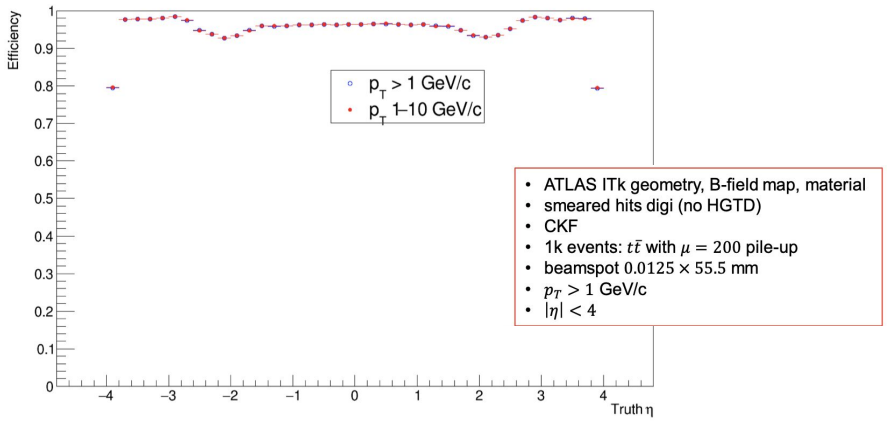
```
if (weight < bottomSP.quality() and weight < middleSP.quality() and
    weight < topSpVec[i]->quality()) {
  continue;
}
```

ACTS-ITk Seeding Performance

ITk seeding configuration incorporated into chained style for Python modules:

- Standalone full chain ITk example → ITk seeding with simple python configuration

Excellent performance when compared to the previous default (generic detector's) seeding configuration




```python
s = addParticleGun(
    s,
    MomentumConfig(1.0 * u.GeV, 10.0 * u.GeV, True),
    EtaConfig(-4.0, 4.0, True),
    ParticleConfig(1, acts.PdgParticle.eMuon, True),
    rnd=rnd,
)
s = addFatras(
    s,
    trackingGeometry,
    field,
    outputDirRoot=outputDir,
    rnd=rnd,
)
s = addDigitization(
    s,
    trackingGeometry,
    field,
    digiConfigFile=geo_dir / "itk-hgtd/itk-smearing-config.json",
    outputDirRoot=outputDir,
    rnd=rnd,
)
s = addSeeding(
    s,
    trackingGeometry,
    field,
    TruthSeedRanges(pt=(1.0 * u.GeV, None), eta=(-4.0, 4.0), nHits=(9, None)),
    *itkSeedingAlgConfig("PixelSpacePoints"),
    geoSelectionConfigFile=geo_dir / "itk-hgtd/geoSelection-ITk.json",
    outputDirRoot=outputDir,
)
s = addCKFTracks(
    s,
    trackingGeometry,
    field,
    CKFPerformanceConfig(ptMin=400.0 * u.MeV, nMeasurementsMin=6),
    outputDirRoot=outputDir,
)
s.run()
```

ITk seeding configuration

Python Example: https://github.com/acts-project/acts/blob/main/Examples/Scripts/Python/full_chain_itk.py

Seeding & Pattern - ACTS Seed Finder

Luis Falda Coelho

ACTS Workshop

Sep 26th, 2022