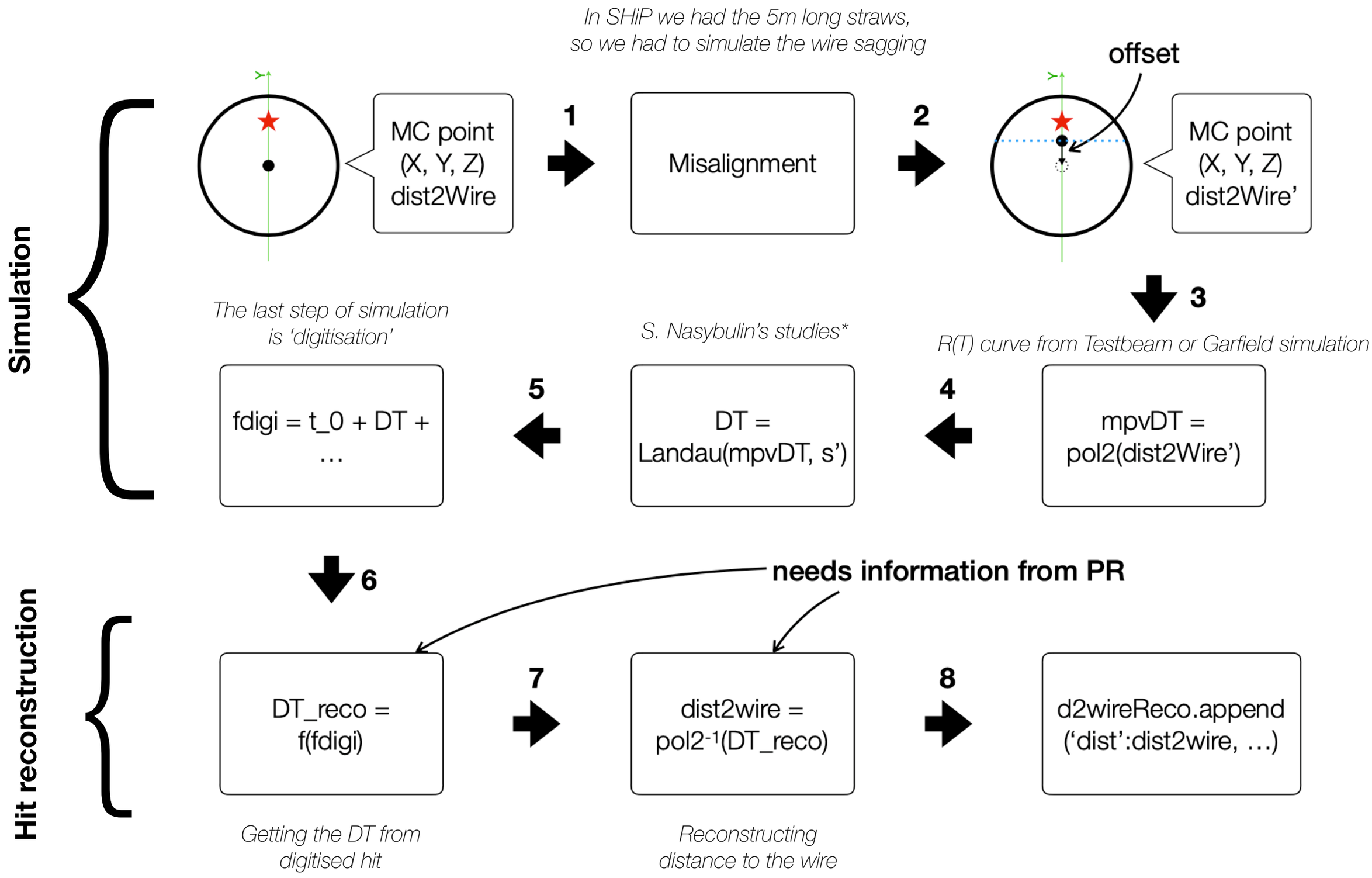
Possible implementation of Garfield-based straw signal parametrisation for realistic simulation of Straw Tracker response

Andrei Zelenov (PNPI)
23/03/2022

- We proposed the straw signal parametrisation in the same way how it was realised in FairSHiP
- MC hits produced by GEANT4 were smeared and digitised according to parametrisation based on Garfield ‘predictions’ and/or Testbeam measurements


Simulation
MC point
(X, Y, Z)
dist2Wire
1
Misalignment
2
offset
MC point
(X, Y, Z)
dist2Wire'
3
R(T) curve from Testbeam or Garfield simulation
mpvDT =
pol2(dist2Wire')
4
DT =
Landau(mpvDT, s')
S. Nasybulin's studies*
5
fdigi = t_0 + DT +
…
The last step of simulation
is 'digitisation'
6
needs information from PR
Hit reconstruction
DT_reco =
f(fdigi)
Getting the DT from
digitised hit
7
dist2wire =
pol2-1(DT_reco)
Reconstructing
distance to the wire
8
d2wireReco.append
('dist':dist2wire, …)
In SHiP we had the 5m long straws,
so we had to simulate the wire sagging

The Drift Time parameterization was made with respect to TestBeam2017 results and special GARFIELD studies(made by S. Nasybulin)

$$t^{MPV} = a \cdot \text{dist2Wire}^2 + b$$

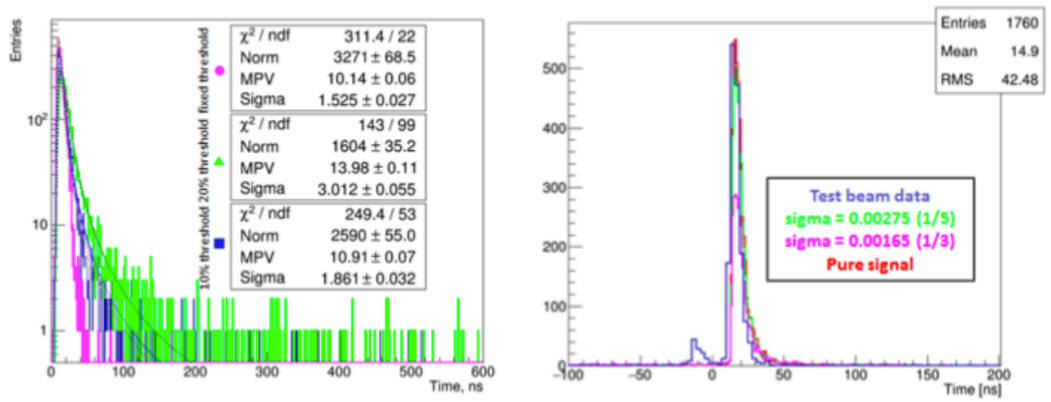
$$\text{DT} = \text{Landau}(t^{MPV}, \sigma_t),$$

where

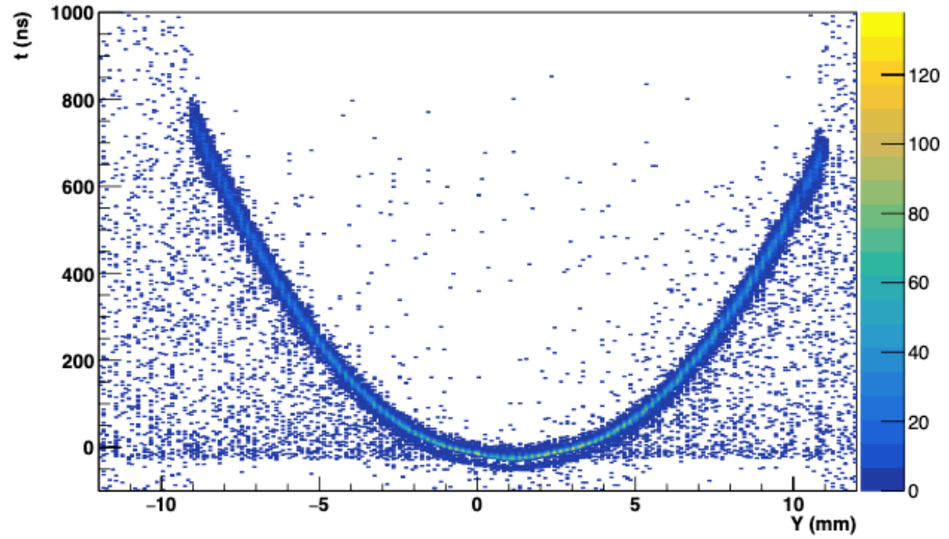
$$\sigma_t(Y) = \left(t^{MPV}/100\right) \cdot \left(8.52 \cdot e^{-0.466 \cdot Y} + 31.81 \cdot e^{-2.392 \cdot Y} + 0.419\right)$$

The main idea of tube/wire sagging is recalculation of the dist2Wire by given value of tube/wire shift

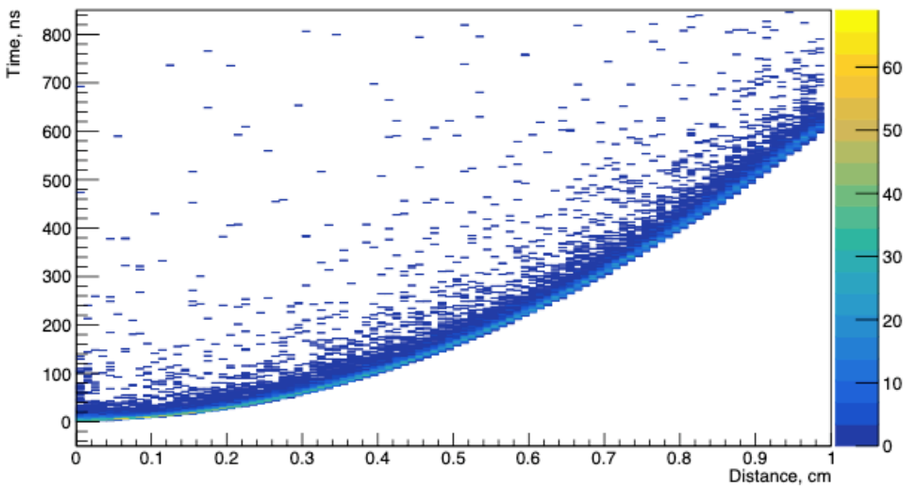
GARFIELD studies



TestBeam2017 DT=f(dist2Wire) relation



Drift Time vs distance to wire



S. Nasybulin's studies (GARFIELD simulations for SPD)

- The same study was made by S. Nasybulin for the straw prototypes, which planned to use in SPD.

- As well as in SHiP case we have a quite good agreement with experimental data

- Based on Garfield simulations results we can provide the parameterisation of straw response (TDC, ADC) for tracks crossing a straw in different magnetic field, with different angles, etc

Fig.1

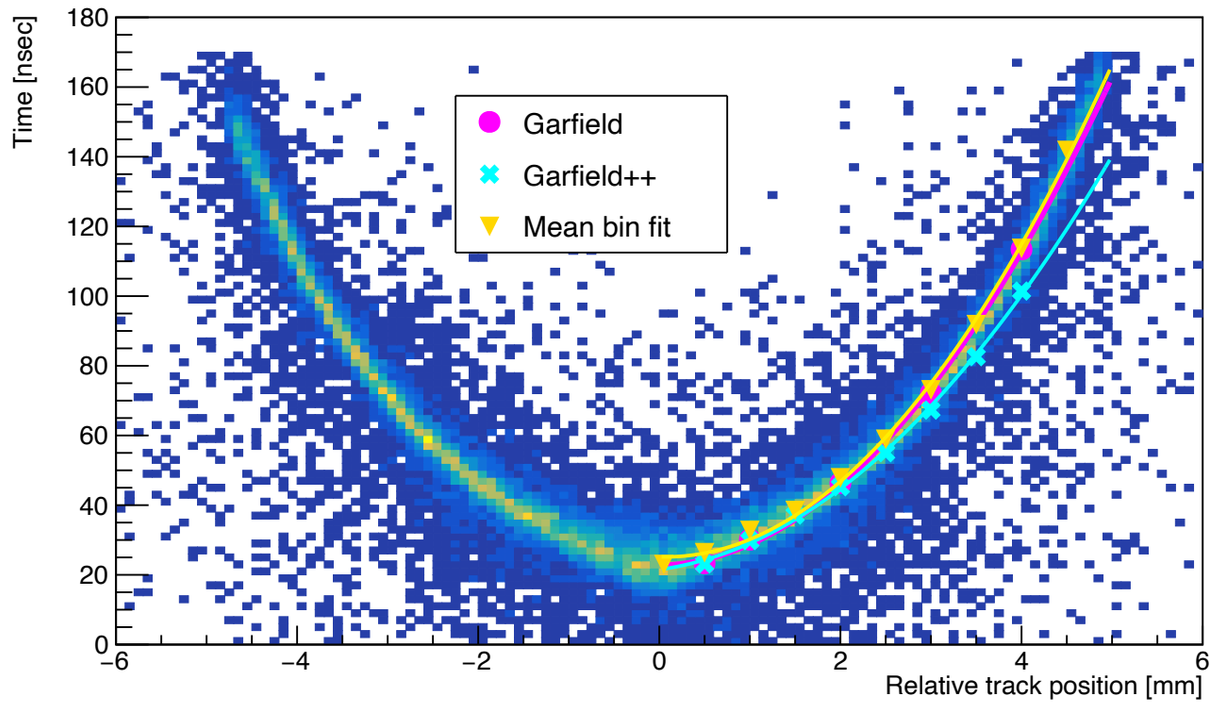
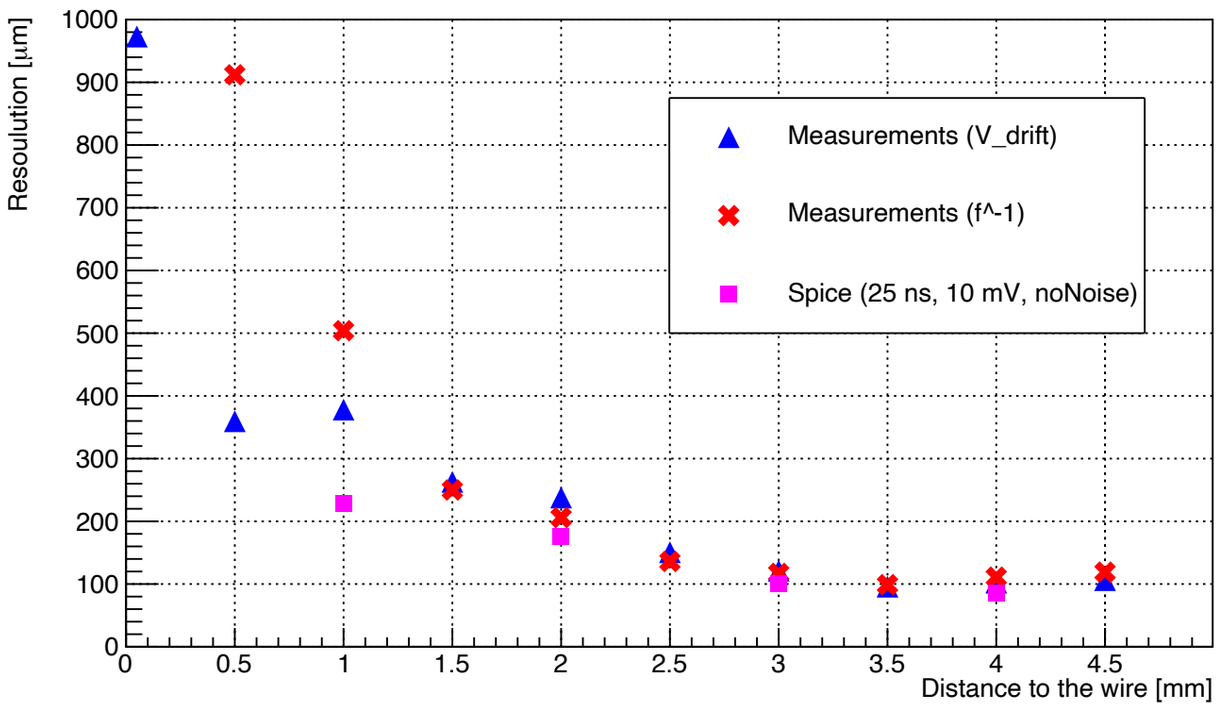
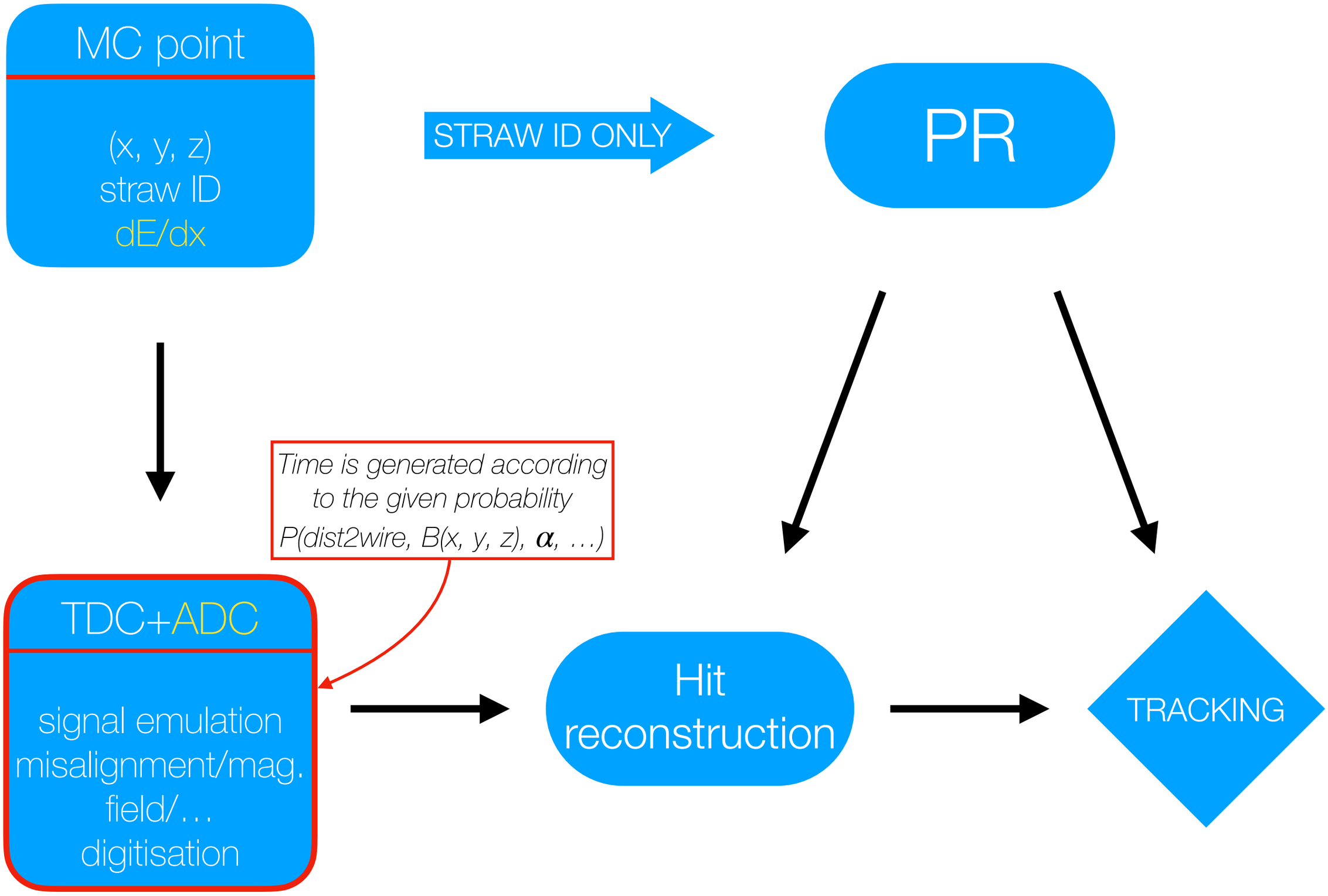


Fig.2



The principal scheme of Simulation → Reconstruction procedure in Straw Tracker

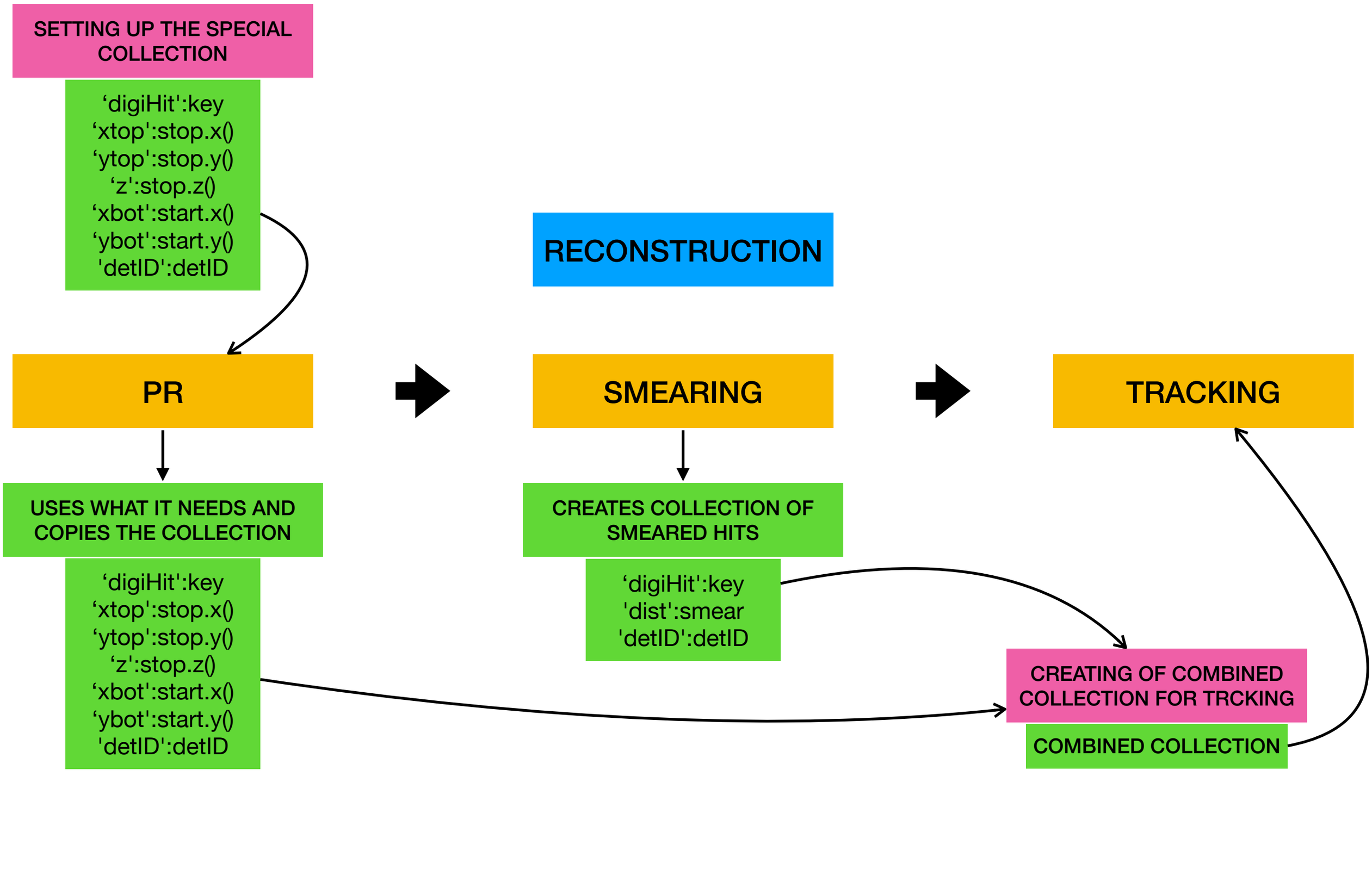


- The first Garfield simulation results are obtained; work on fine tuning is ongoing

- Comparison to Testbeam results is necessary for accurate crosscheck

- The work on complex parameterisation taking into account various parameters, such as magnetic field, readout electronics characteristics, etc. will require some time

Backup


SETTING UP THE SPECIAL COLLECTION
‘digiHit':key
‘xtop':stop.x()
‘ytop':stop.y()
‘z':stop.z()
‘xbot':start.x()
‘ybot':start.y()
'detID':detID
RECONSTRUCTION
PR
SMEARING
TRACKING
USES WHAT IT NEEDS AND COPIES THE COLLECTION
‘digiHit':key
‘xtop':stop.x()
‘ytop':stop.y()
‘z':stop.z()
‘xbot':start.x()
‘ybot':start.y()
'detID':detID
CREATES COLLECTION OF SMEARED HITS
‘digiHit':key
'dist':smear
'detID':detID
CREATING OF COMBINED COLLECTION FOR TRCKING
COMBINED COLLECTION

```
def smearHits(self,no_amb=None):
. . .
  for aDigi in self.digiStraw:
     . . .
     if (ROOT.strawtubesDigi.Instance().IsDefaultDriftTime()):
         smear = (aDigi.GetDigi() - self.sTree.t0 - p.GetTime() - ( stop[0]-p.GetX() )/ u.speedOfLight) * v_drift
     else:
         TDC = aDigi.GetDigi()
         t0 = self.sTree.t0 + p.GetTime()
         signalPropagationTime = (stop[0]-p.GetX()) / u.speedOfLight
         driftTime = ROOT.strawtubesDigi.Instance().DriftTimeFromTDC(TDC, t0, signalPropagationTime)
         if driftTime < 5.285: driftTime = 5.285
         smear = ROOT.strawtubesDigi.Instance().NewDist2WireFromDriftTime(driftTime)
     if smear > ShipGeo.strawtubes.InnerStrawDiameter: aDigi.setInvalid()
     if no_amb: smear = p.dist2Wire()
     SmearedHits.append( {'digiHit':key, 'dist':smear, 'detID':detID} )
  return SmearedHits
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
  def pseudoCollectionEstimation(self):
    . . .
    for aDigi in self.digiStraw:
        . . .
        pseudoCollection.append( {'digiHit':key,'xtop':stop.x(),'ytop':stop.y(),'z':stop.z(),'xbot':start.x(),'ybot':start.y(),
'detID':detID} )
    return pseudoCollection
```

```
def findTracks(self):
  . . .
  pseudoCollection = self.pseudoCollectionEstimation()
  . . .
  if realPR:
      track_hits = shipPatRec.execute(pseudoCollection, ShipGeo, realPR)
      . . .
      for i_track in track_hits.keys():
         atrack = track_hits[i_track]
         . . .
atrack_stereo34 = atrack['stereo34']
atrack_smeared_hits = list(atrack_y12) + list(atrack_stereo12) + list(atrack_y34) + list(atrack_stereo34)
if withT0:  self.SmearedHits = self.withT0Estimate()
# old procedure, not including estimation of t0
else:       self.SmearedHits = self.smearHits(withNoStrawSmearing)
counter = 0
for sm in atrack_smeared_hits:
   . . .
   m = array('d',[sm['xtop'],sm['ytop'],sm['z'],sm['xbot'],sm['ybot'],sm['z'],self.SmearedHits[counter]['dist']])
   hitPosLists[trID].push_back(ROOT.TVectorD(7,m))
```

Vonelez/FairShip strawtubesDigi branch

```
def findTracks(self):
  . . .
  pseudoCollection = self.pseudoCollectionEstimation()
  . . .
  if realPR:
     track_hits = shipPatRec.execute(pseudoCollection, ShipGeo, realPR)
      . . .
     for i_track in track_hits.keys():
        atrack = track_hits[i_track]
        . . .
```

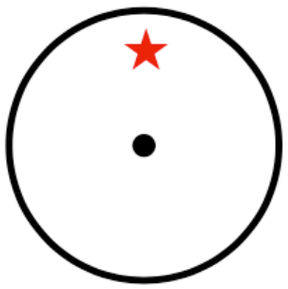
Special collection for PR

PR

```
atrack_stereo34 = atrack['stereo34']
atrack_smeared_hits = list(atrack_y12) + list(atrack_stereo12) + list(atrack_y34) + list(atrack_stereo34)
if withT0:  self.SmearedHits = self.withT0Estimate()
else:       self.SmearedHits = self.smearHits(withNoStrawSmearing)
counter = 0
for sm in atrack_smeared_hits:
   . . .
  m = array('d',[sm['xtop'],sm['ytop'],sm['z'],sm['xbot'],sm['ybot'],sm['z'],self.SmearedHits[counter]['dist']])
  hitPosLists[trID].push_back(ROOT.TVectorD(7,m))
```

Smearing

Setting up the collection for tracking



MC point
(X, Y, Z)
dist2Wire

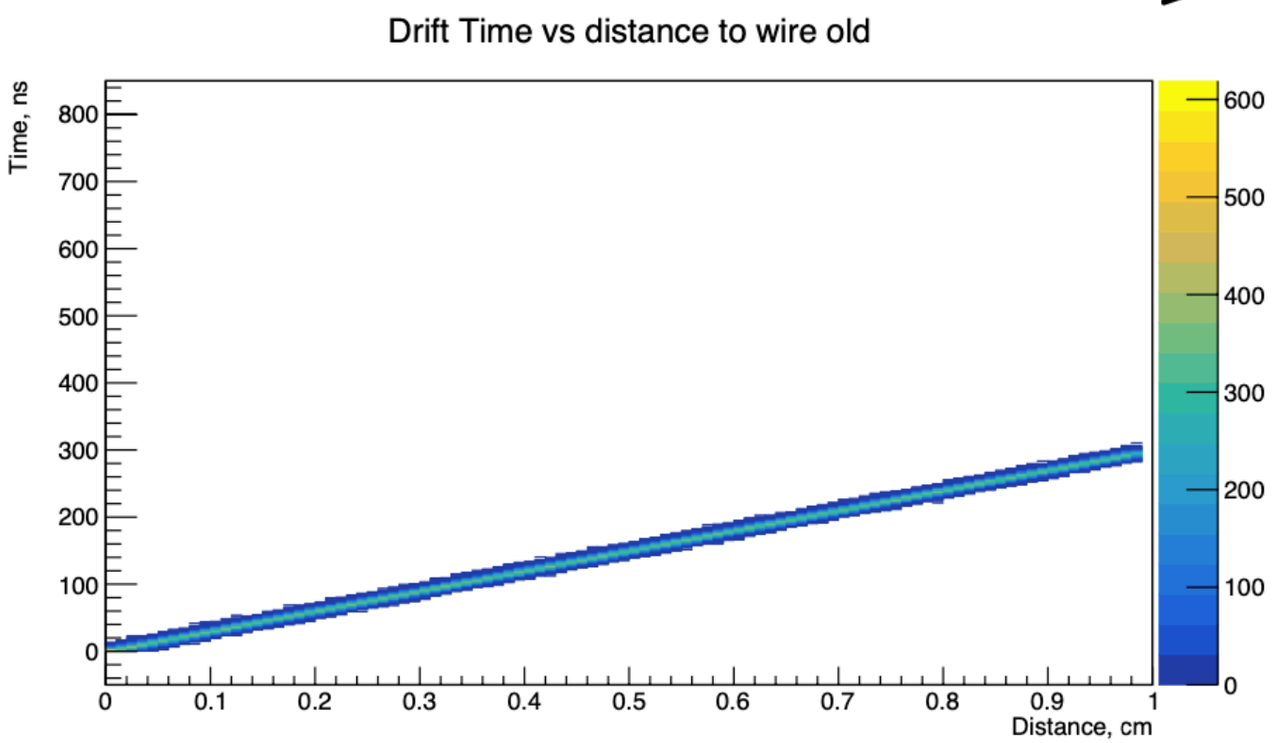
1

DT =
Gaus(dist2Wire, s)
/ v_drift

2

fdigi = t_0 + DT +
…

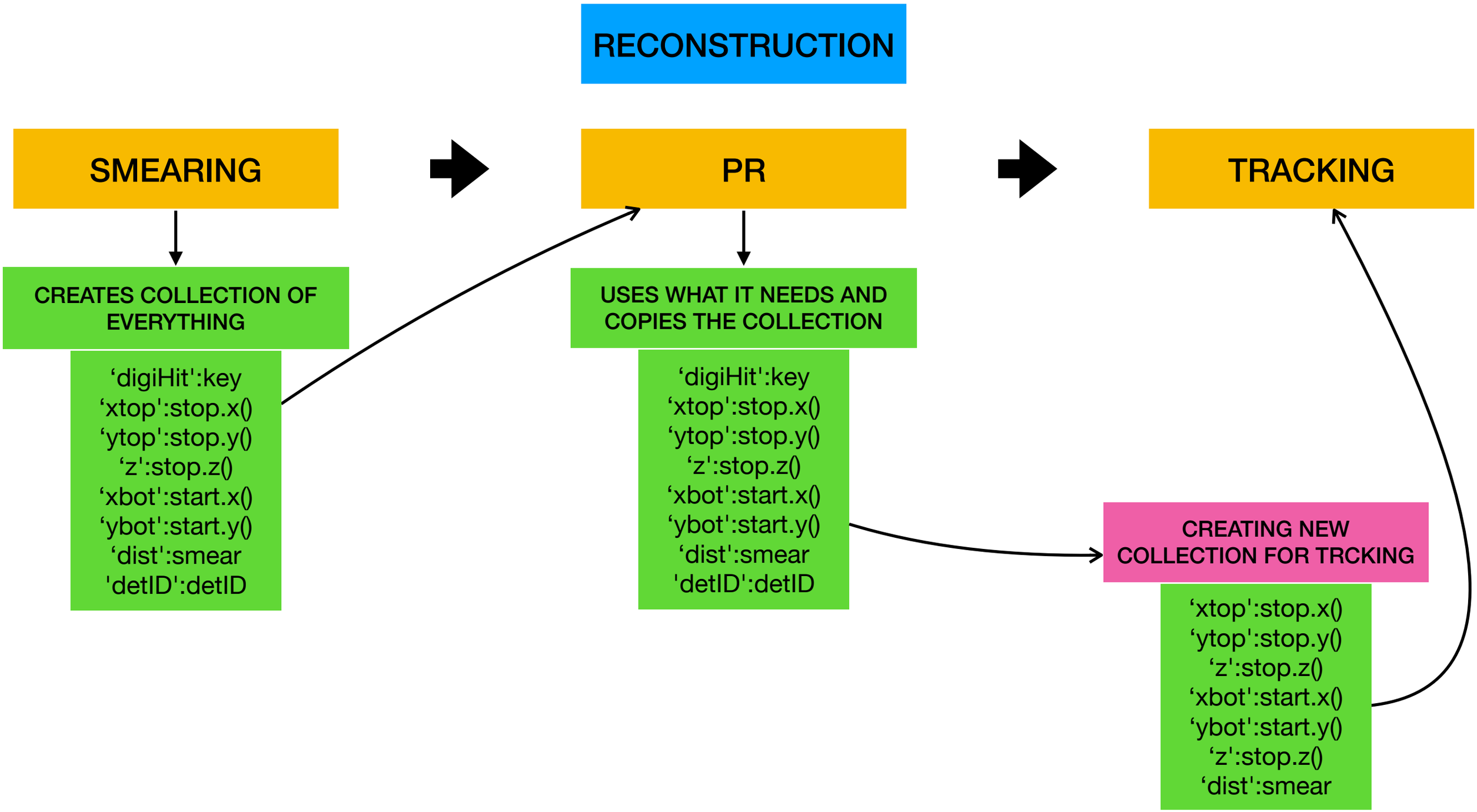
3



smear =
(fdigi - t_0 - …) *
v_drift

4

smearHits.append
(‘dist’:smear, …)


RECONSTRUCTION
SMEARING
PR
TRACKING
CREATES COLLECTION OF EVERYTHING
USES WHAT IT NEEDS AND COPIES THE COLLECTION
‘digiHit':key
‘xtop':stop.x()
‘ytop':stop.y()
‘z':stop.z()
‘xbot':start.x()
‘ybot':start.y()
‘dist':smear
'detID':detID
‘digiHit':key
‘xtop':stop.x()
‘ytop':stop.y()
‘z':stop.z()
‘xbot':start.x()
‘ybot':start.y()
‘dist':smear
'detID':detID
CREATING NEW COLLECTION FOR TRCKING
‘xtop':stop.x()
‘ytop':stop.y()
‘z':stop.z()
‘xbot':start.x()
‘ybot':start.y()
‘z':stop.z()
‘dist':smear

```
def smearHits(self,no_amb=None):

. . .

  for aDigi in self.digiStraw:

    . . .

    smear = (aDigi.GetDigi() - self.sTree.t0  - p.GetTime() - ( stop[0]-p.GetX() )/ u.speedOfLight) * v_drift

    if no_amb: smear = p.dist2Wire()

    SmearedHits.append( {'digiHit':key,'xtop':stop.x(),'ytop':stop.y(),'z':stop.z(),'xbot':start.x(),'ybot':start.y(),'dist':smear,
'detID':detID} )

   return SmearedHits
```

```
def findTracks(self):

if withT0:  self.SmearedHits = self.withT0Estimate()
else:       self.SmearedHits = self.smearHits(withNoStrawSmearing)
if realPR:
   track_hits = shipPatRec.execute(self.SmearedHits, ShipGeo, realPR)
   for i_track in track_hits.keys():
       atrack = track_hits[i_track]
       . . .
       atrack_stereo34 = atrack['stereo34']
       atrack_smeared_hits = list(atrack_y12) + list(atrack_stereo12) + list(atrack_y34) + list(atrack_stereo34)
       for sm in atrack_smeared_hits:
          . . .
          m = array('d',[sm['xtop'],sm['ytop'],sm['z'],sm['xbot'],sm['ybot'],sm['z'],sm['dist']])
          hitPosLists[trID].push_back(ROOT.TVectorD(7,m))
```

```
  def smearHits(self,no_amb=None):

  . . .

    for aDigi in self.digiStraw:

       . . .

       smear = (aDigi.GetDigi() - self.sTree.t0  - p.GetTime() - ( stop[0]-p.GetX() )/ u.speedOfLight) * v_drift

       if no_amb: smear = p.dist2Wire()

       SmearedHits.append( {'digiHit':key,'xtop':stop.x(),'ytop':stop.y(),'z':stop.z(),'xbot':start.x(),'ybot':start.y(),'dist':smear,
'detID':detID} )

     return SmearedHits
```

```
  def findTracks(self):

if withT0:   self.SmearedHits = self.withT0Estimate()
else:        self.SmearedHits = self.smearHits(withNoStrawSmearing)
if realPR:
   track_hits = shipPatRec.execute(self.SmearedHits, ShipGeo, realPR)
   for i_track in track_hits.keys():
       atrack = track_hits[i_track]
       . . .
       atrack_stereo34 = atrack['stereo34']
       atrack_smeared_hits = list(atrack_y12) + list(atrack_stereo12) + list(atrack_y34) + list(atrack_stereo34)
       for sm in atrack_smeared_hits:
          . . .
          m = array('d',[sm['xtop'],sm['ytop'],sm['z'],sm['xbot'],sm['ybot'],sm['z'],sm['dist']])
          hitPosLists[trID].push_back(ROOT.TVectorD(7,m))
```

Smearing

PR

Setting up the collection for tracking