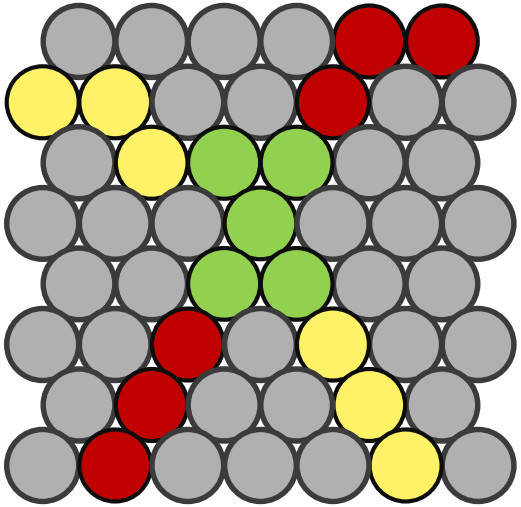
HOW TO SPEED UP THE HOUGH TRACK FINDER

27.10.2020 I PANDA COLLABORATION MEETING | ANNA ALICKE

HOUGH TRACK FINDER

Review: basic procedure

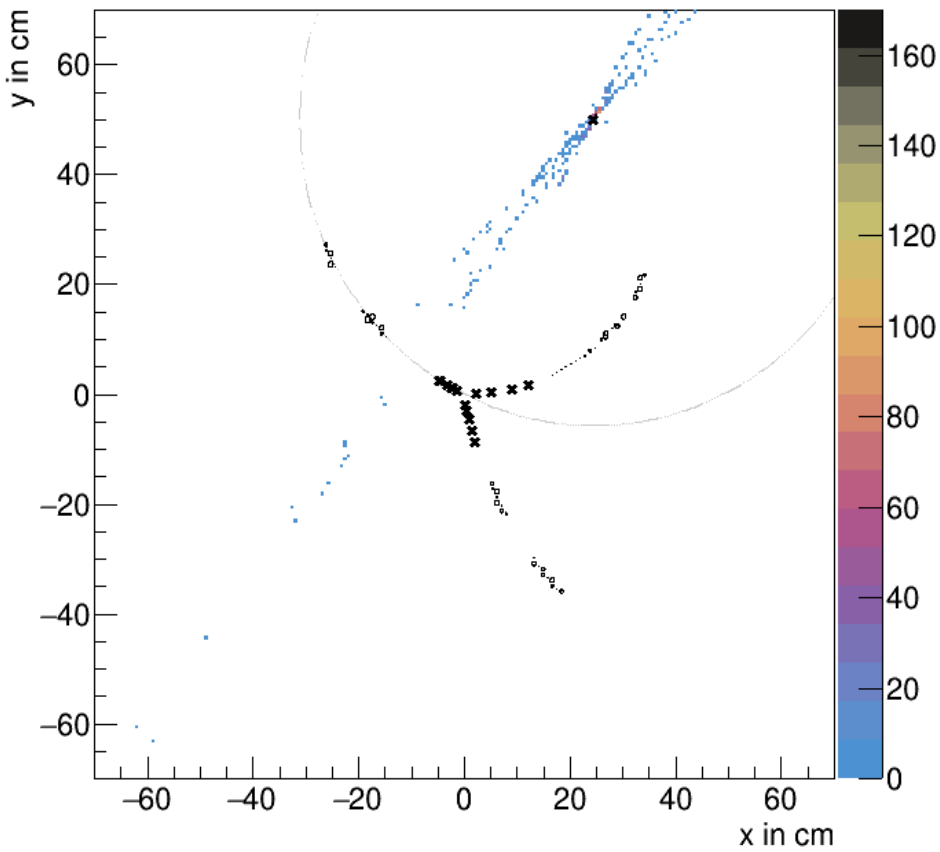
- Generate tracklets
  - Cellular automaton
  - Segmentation



HOUGH TRACK FINDER

Review: basic procedure

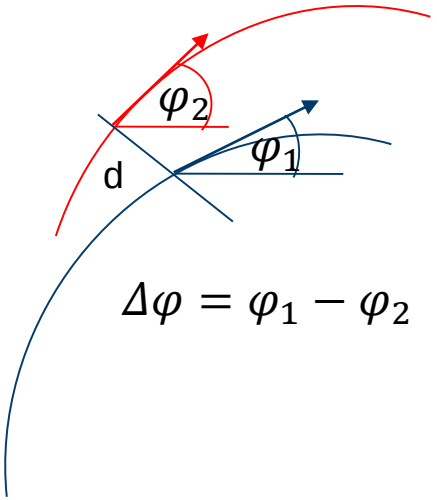
- Generate tracklets
  - Cellular automaton
  - Segmentation
- Find circle for each tracklet
  - Hough transformation



HOUGH TRACK FINDER

Review: basic procedure

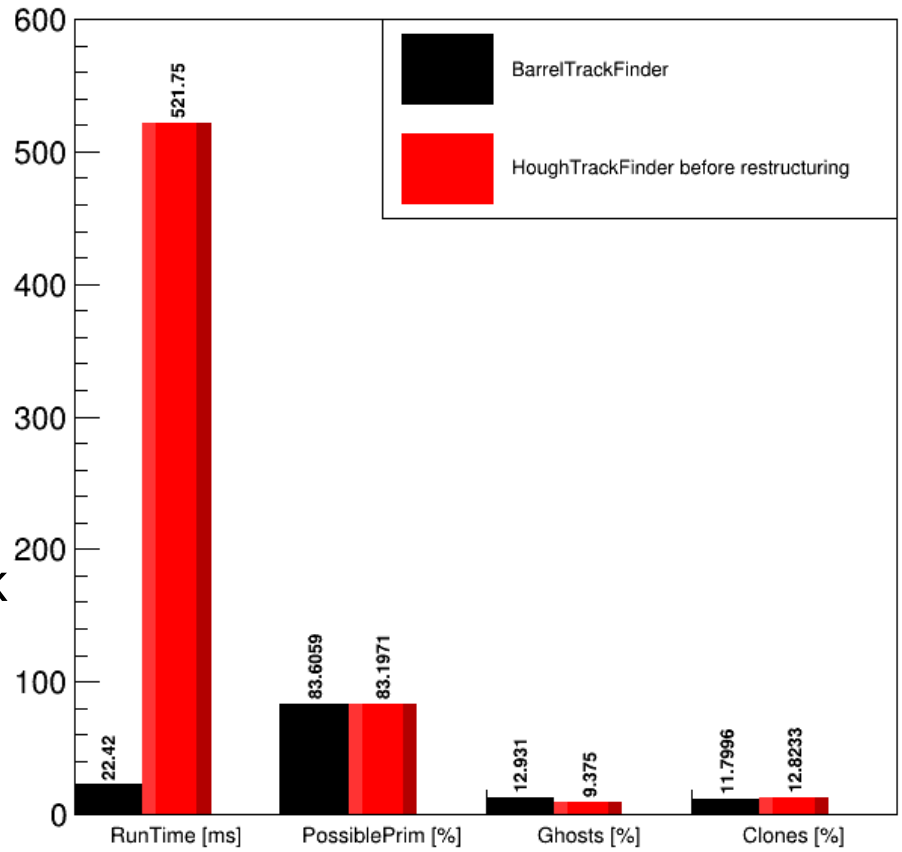
- Generate tracklets
  - Cellular automaton
  - Segmentation
- Find circle for each tracklet
  - Hough transformation
- Merging
  - combine tracklets to particle track



HOUGH TRACK FINDER

Review: basic procedure

- Generate tracklets
  - Cellular automaton
  - Segmentation
- Find circle for each tracklet
  - Hough transformation
- Merging
  - combine tracklets to particle track



Too slow!

MAIN CHANGES TO SPEED UP THE CODE

- Generate tracklets
  - Cellular automaton
  - Segmentation

New data structure for hits

- Find circle for each tracklet
  - Hough transformation

New data structure for hough space

- Merging
  - combine tracklets to particle track

MAIN CHANGES TO SPEED UP THE CODE

New data structure for hits

Similar to CellularAutomaton
map FairLink to the relevant data:

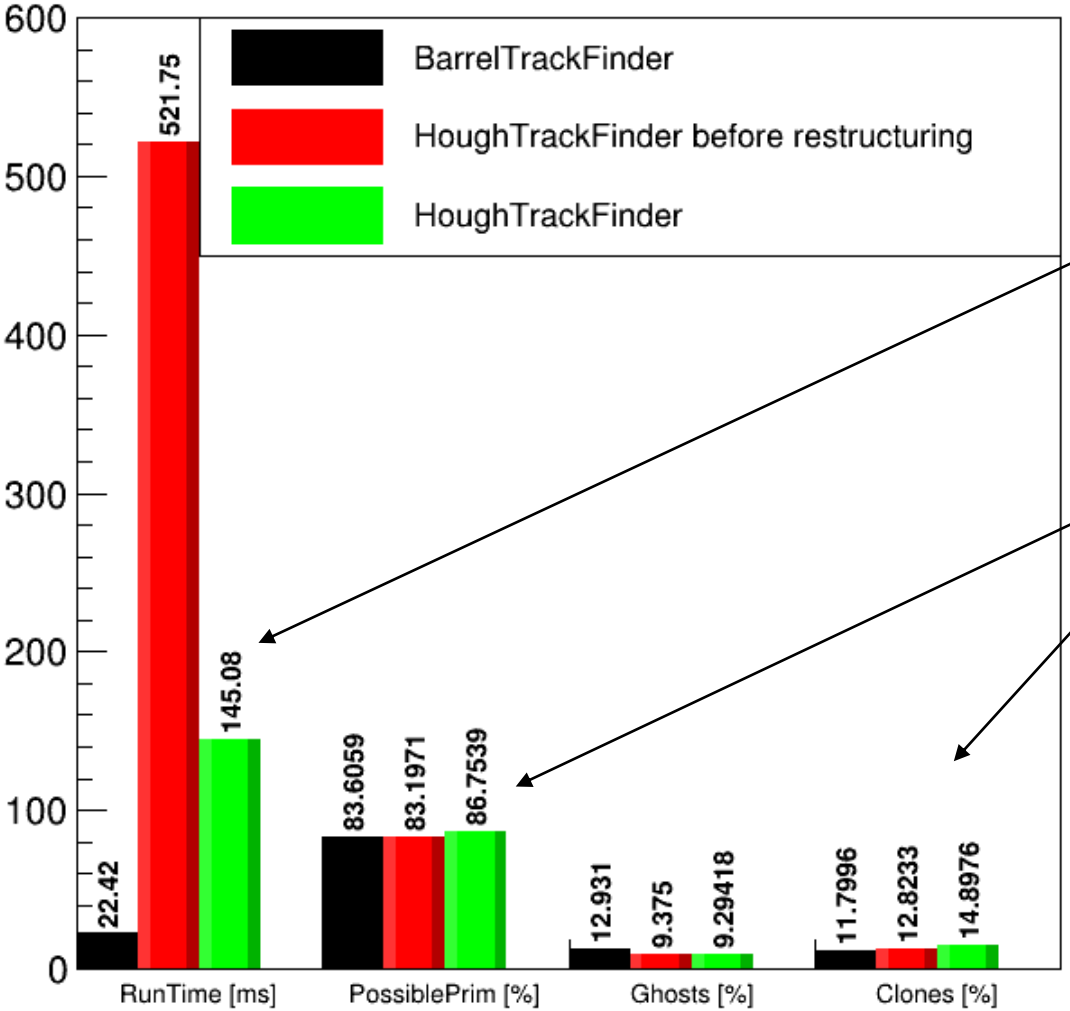
- FairLink - FairHit
- FairLink - Isochrone
- FairLink - Isochrone error
- FairLink - Tube Id
- Tube - Hit
- FairLink - GEM neighbors
- FairLink - STT neighbors

→ faster than typecasting

New data structure for hough space

HoughSpace: map bin to entries

MAIN CHANGES TO SPEED UP THE CODE



Speed up by a factor of 3.5
But still slow

Small change in binning:
Doesn‘t affect the result

- efficiency increases a bit
- but clones also increase a bit

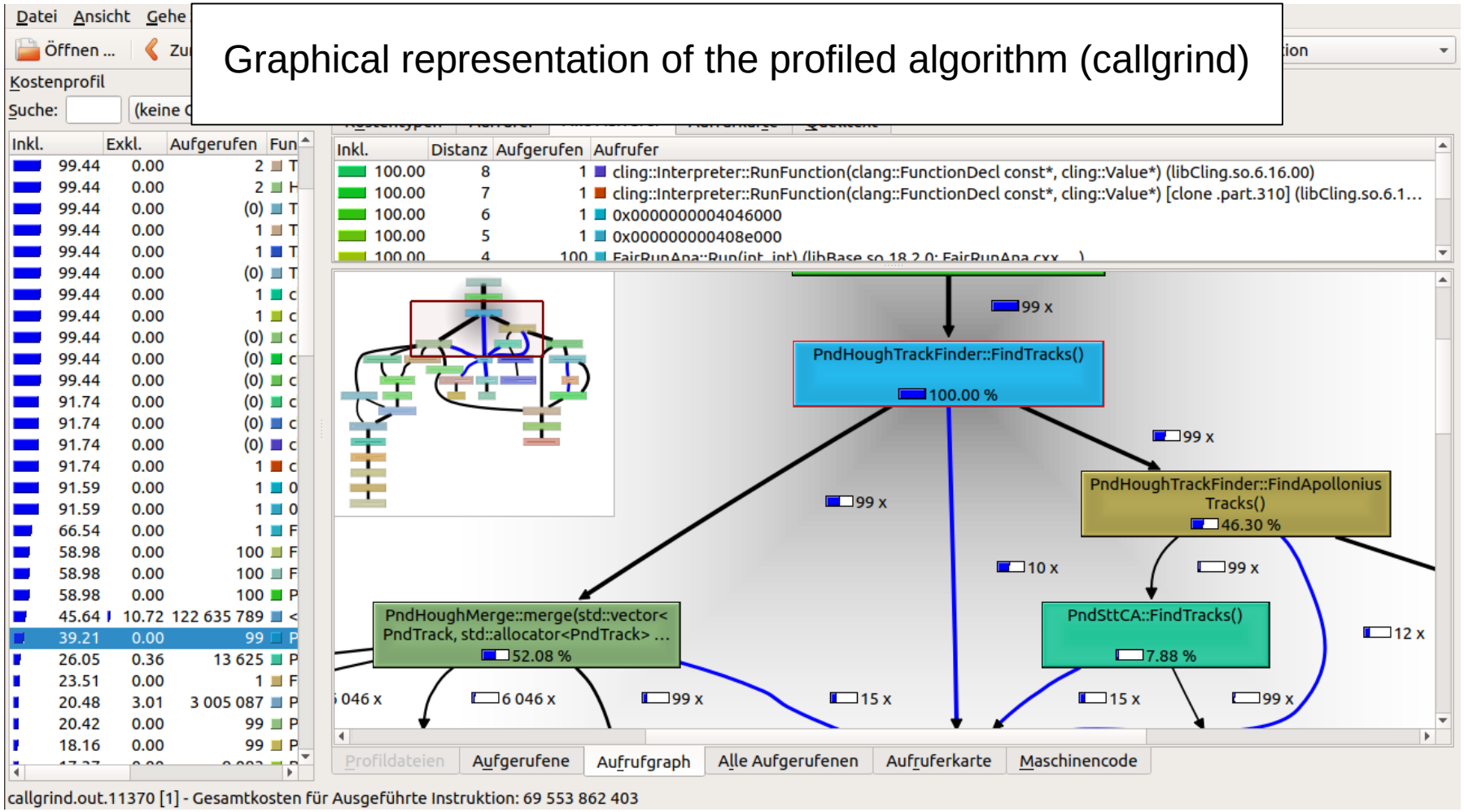
JÜLICH Forschungszentrum

VALGRIND

Further analysis with valgrind

Valgrind is a profiling tool to analyse where the code spends the most time

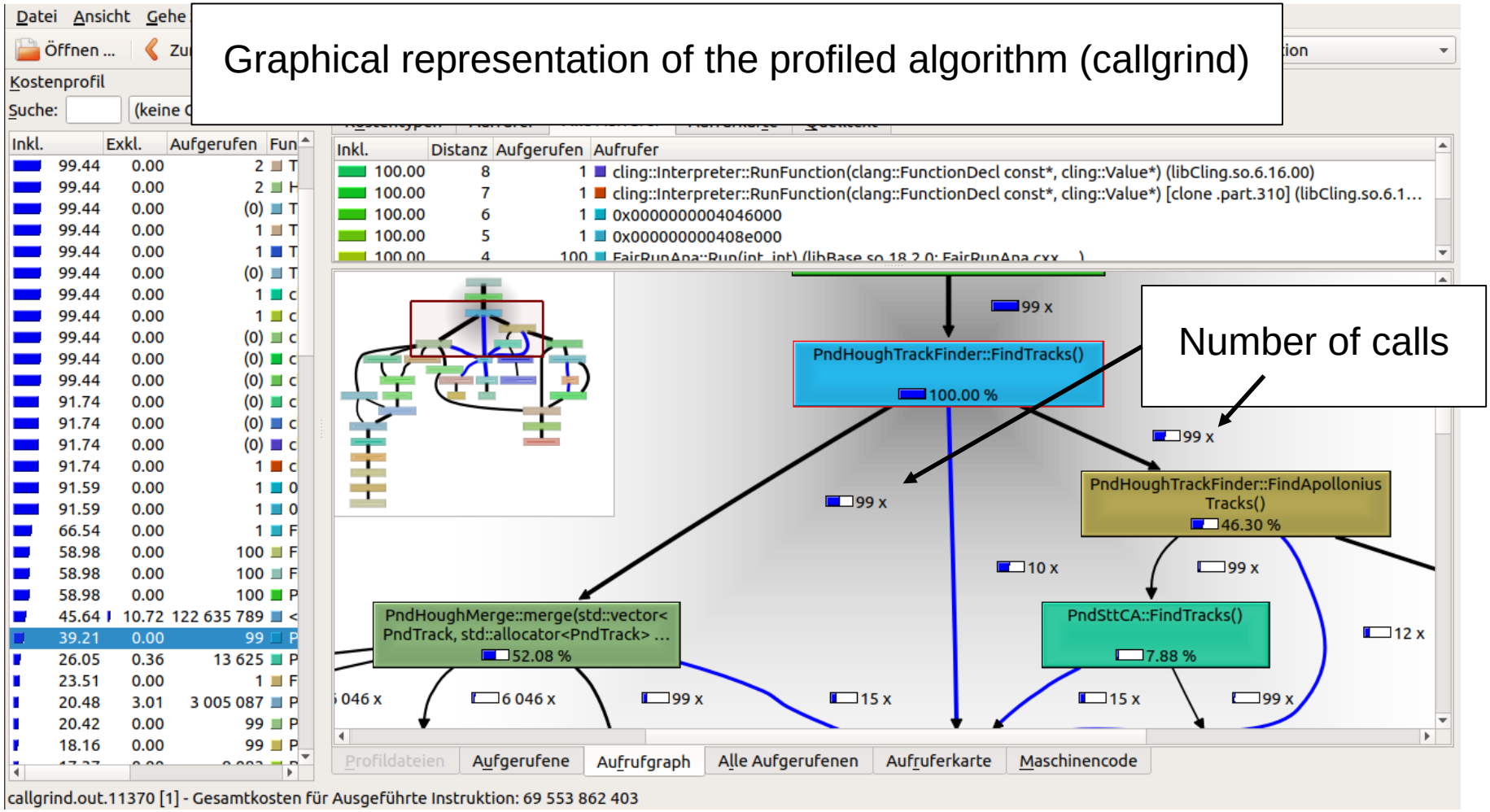
Graphical representation of the profiled algorithm (callgrind)



VALGRIND

Further analysis with valgrind

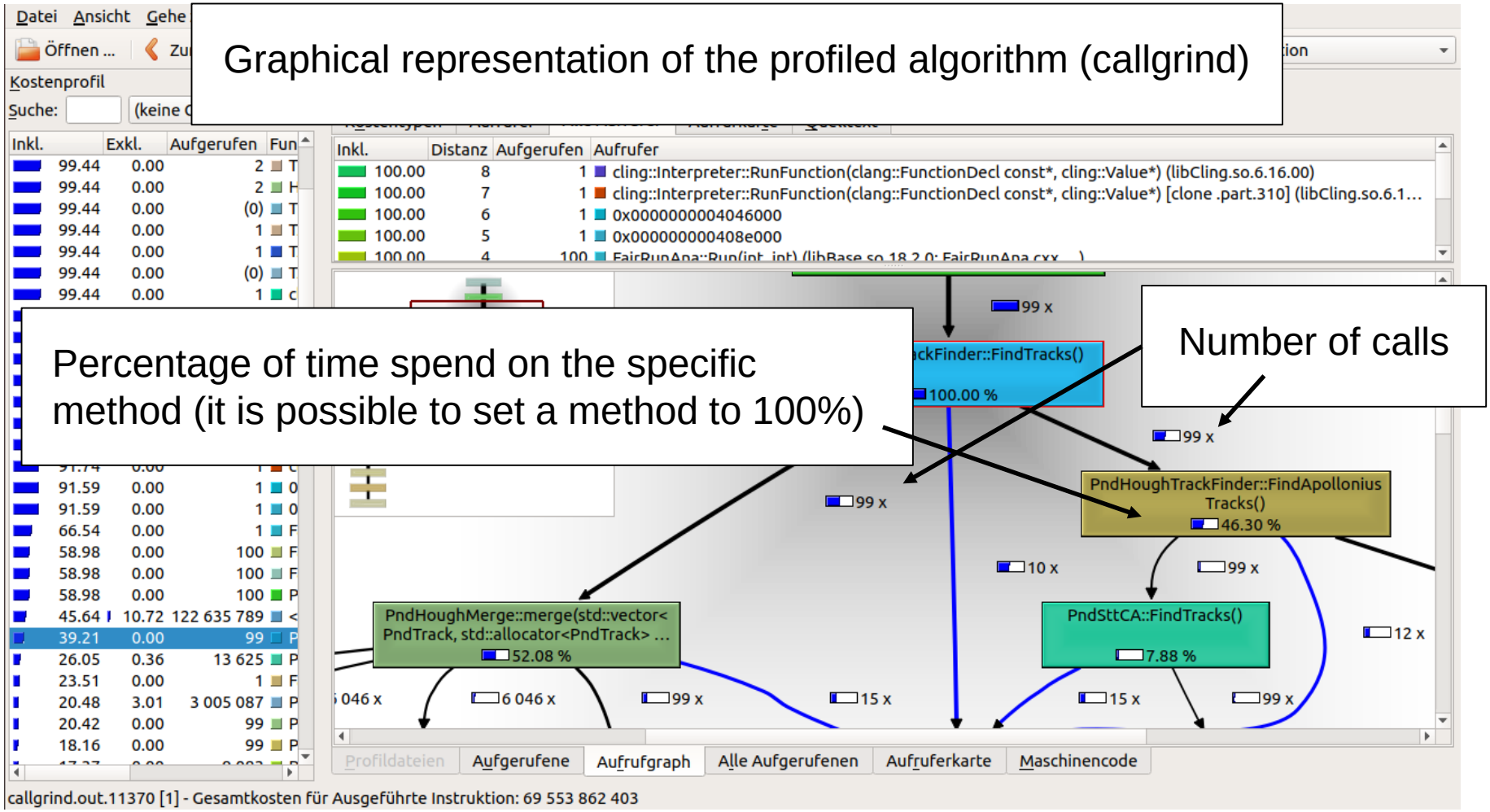
Valgrind is a profiling tool to analyse where the code spends the most time



VALGRIND

Further analysis with valgrind

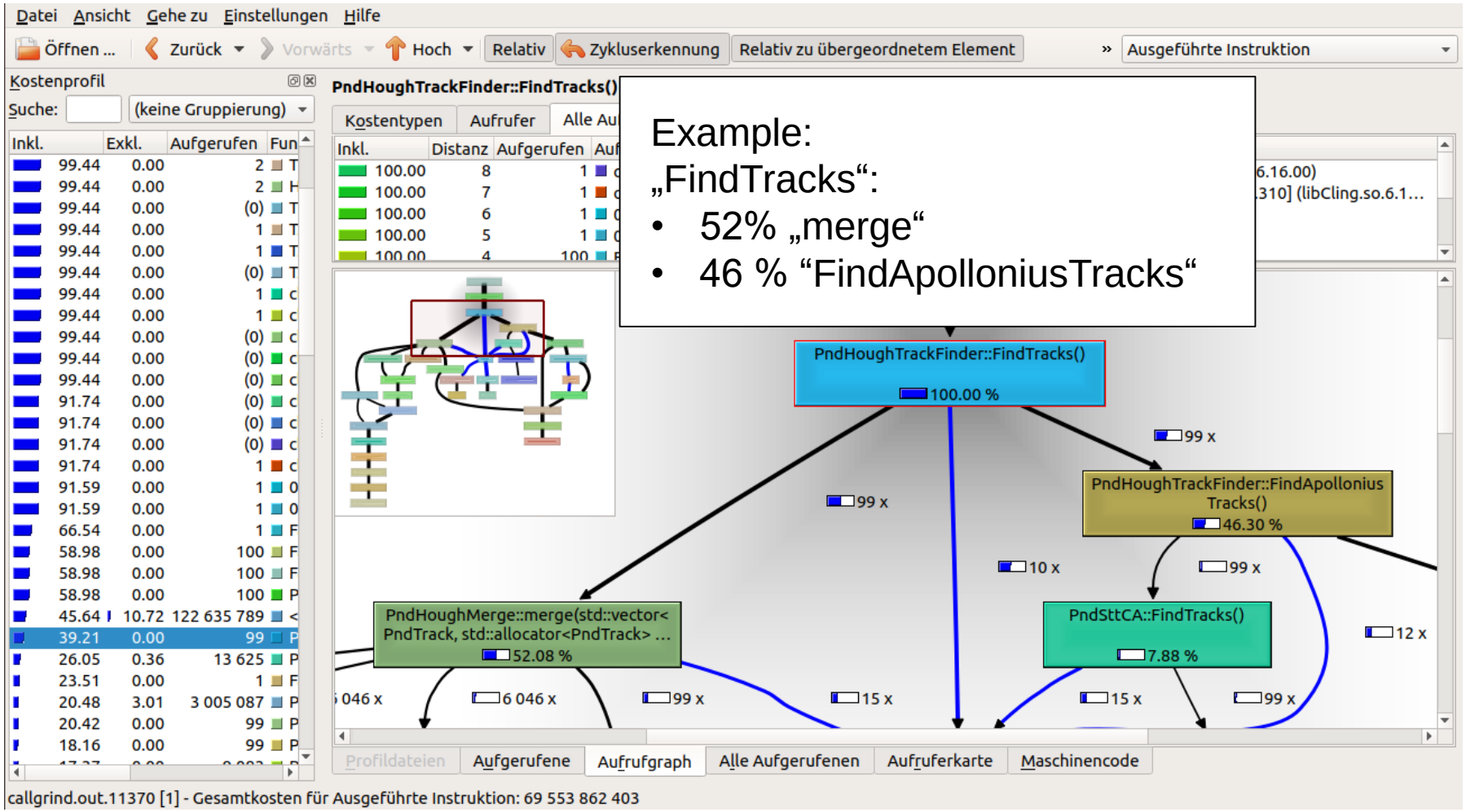
Valgrind is a profiling tool to analyse where the code spends the most time



VALGRIND

Further analysis with valgrind

Valgrind is a profiling tool to analyse where the code spends the most time



Example:
„FindTracks“:

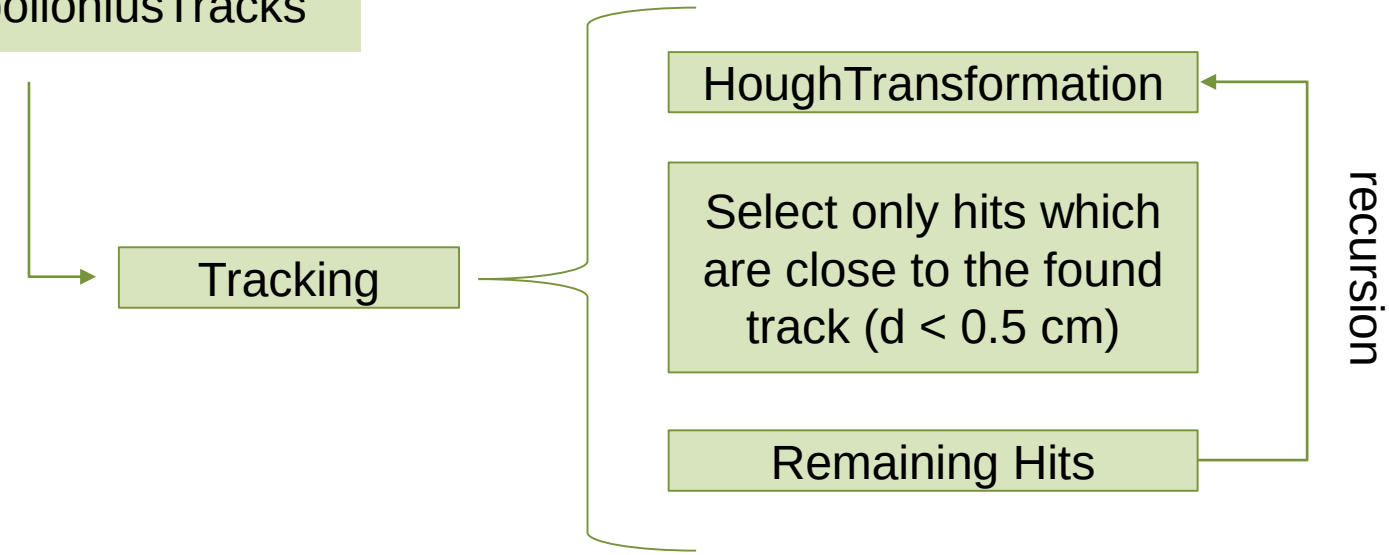
- 52% „merge“
- 46 % “FindApolloniusTracks“

VALGRIND

Speed up tracking

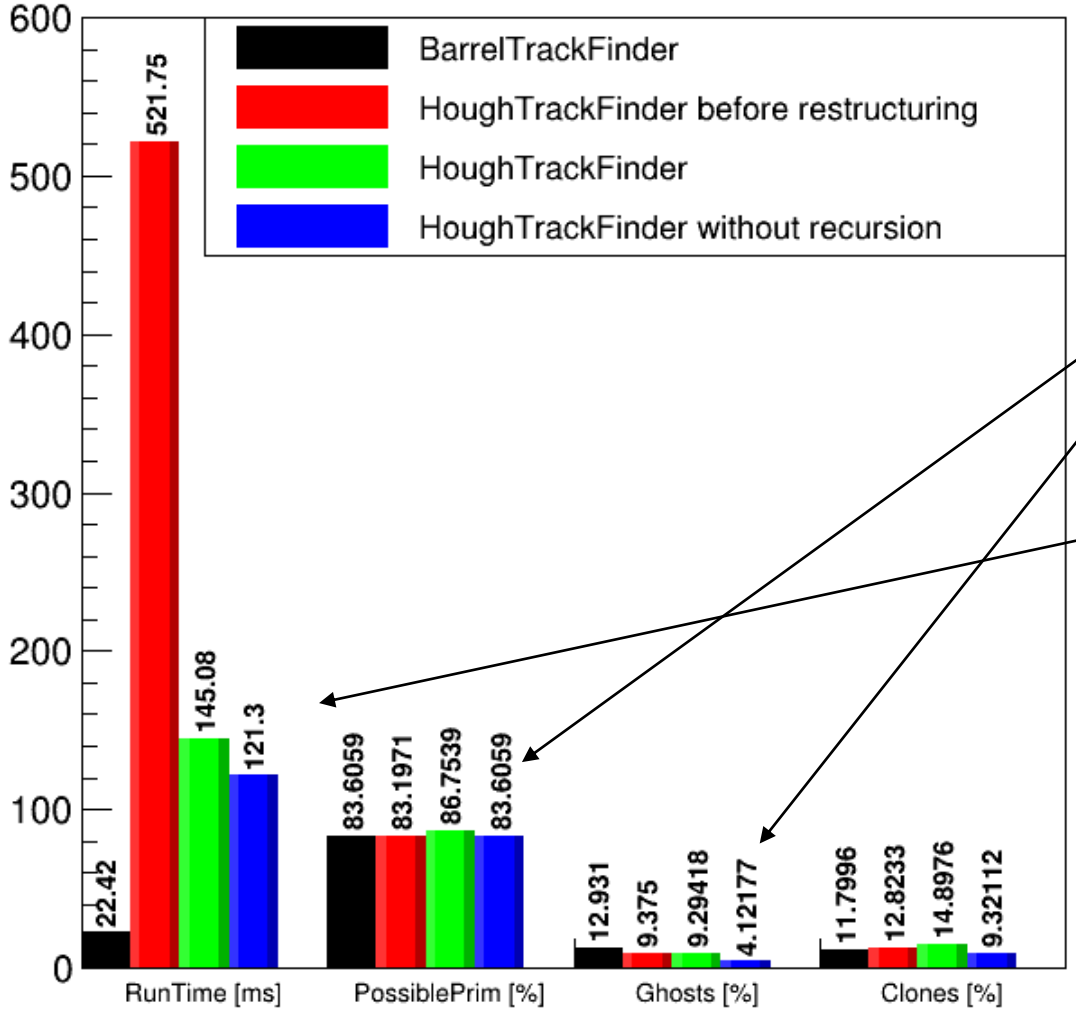
„FindTracks“:

- 46 % “FindApolloniusTracks“
- 52% „merge“



→ How large is the influence of the recursion?
(should be small because of preselection method)

HOUGH TRACK FINDER WITHOUT RECURSION



Efficiency still high

Runtime a bit better

JÜLICH Forschungszentrum

VALGRIND

Speed up merge

„FindTracks“:

- 46 % “FindApolloniusTracks“
- 52% „merge“

Calculate d	19 %
Calculate phi	19 %
Merge single tracklets	34 %

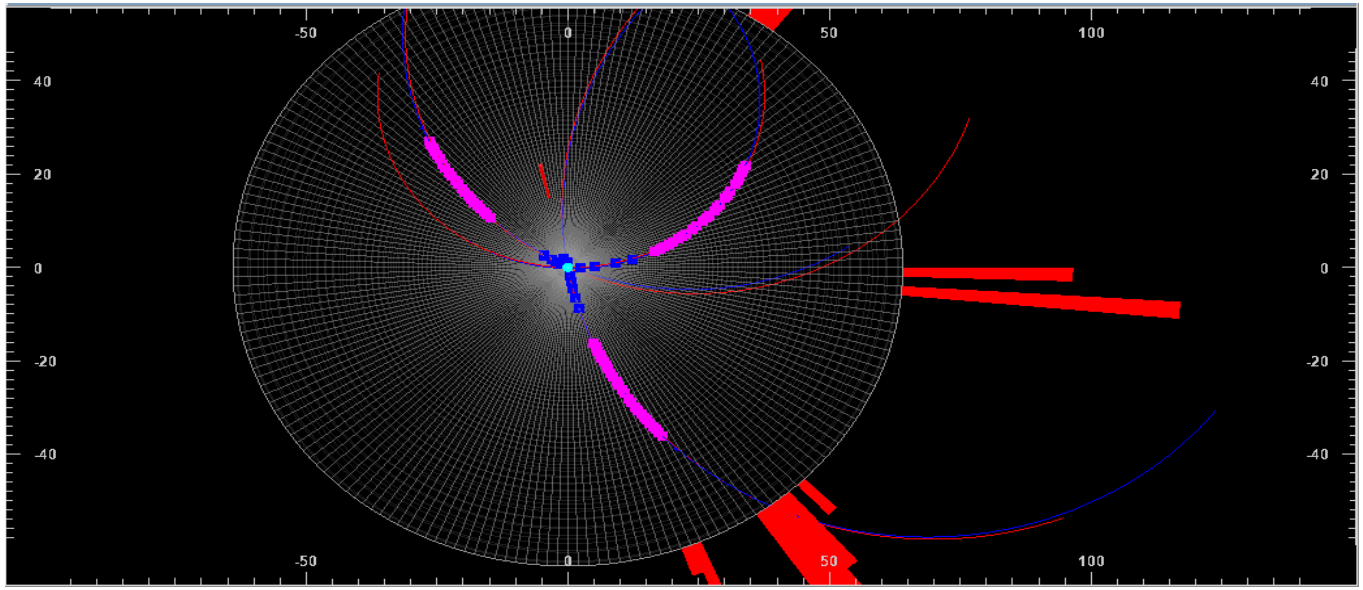
calculate all merging parameters takes a similarly amount of time

→ find an in general better merging algorithm

SPEED UP MERGE

Trying to find a better handling of the Hough transformation and merging to reduce the runtime:

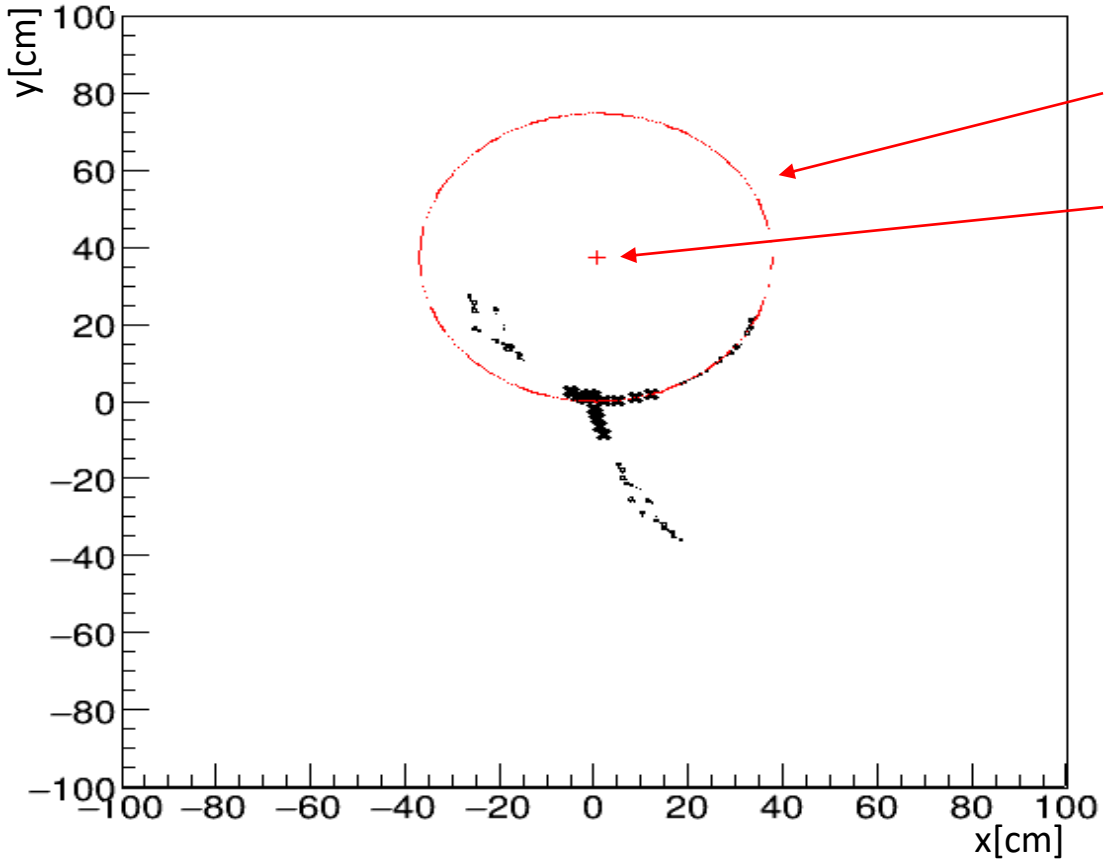
→ investigated correlations in Hough space



SPEED UP MERGE

Trying to find a better handling of the Hough transformation and merging to reduce the runtime:

→ investigated correlations in Hough space



MC Track

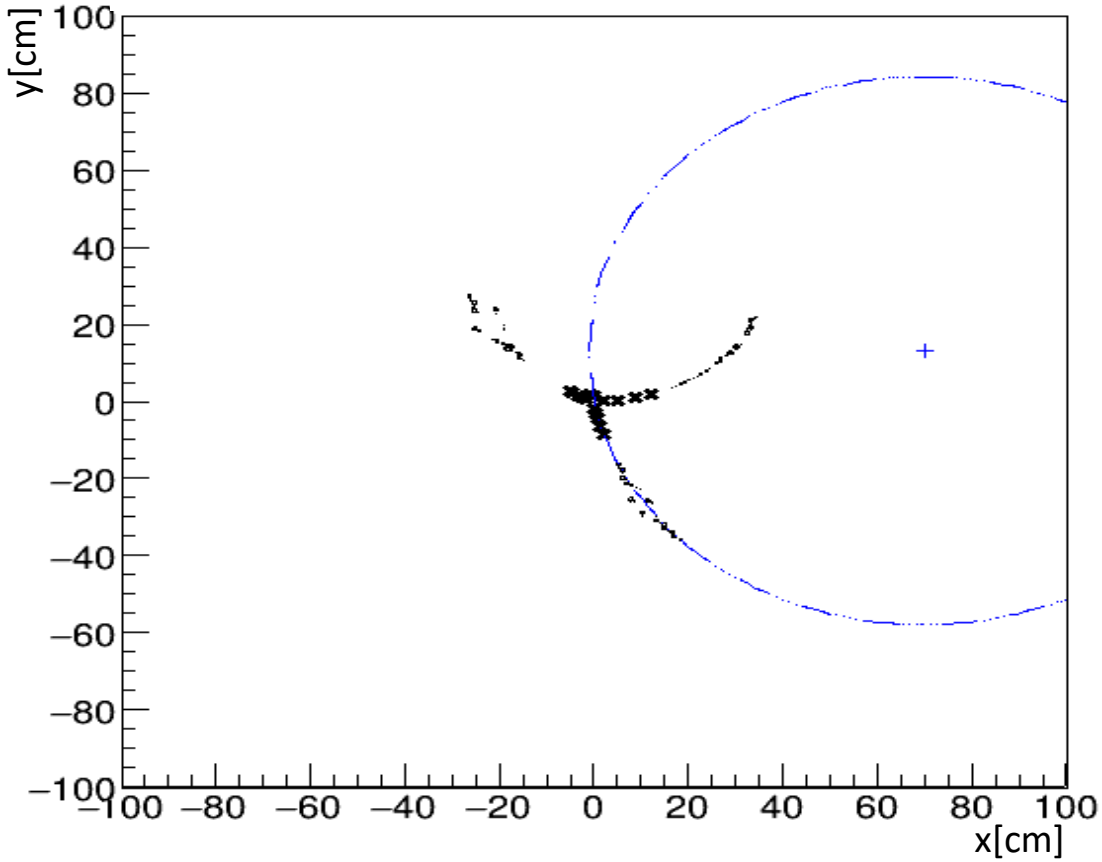
Center of MC Track
➔ MC entry in Hough space

\+ MC track 1

SPEED UP MERGE

Trying to find a better handling of the Hough transformation and merging to reduce the runtime:

→ investigated correlations in Hough space



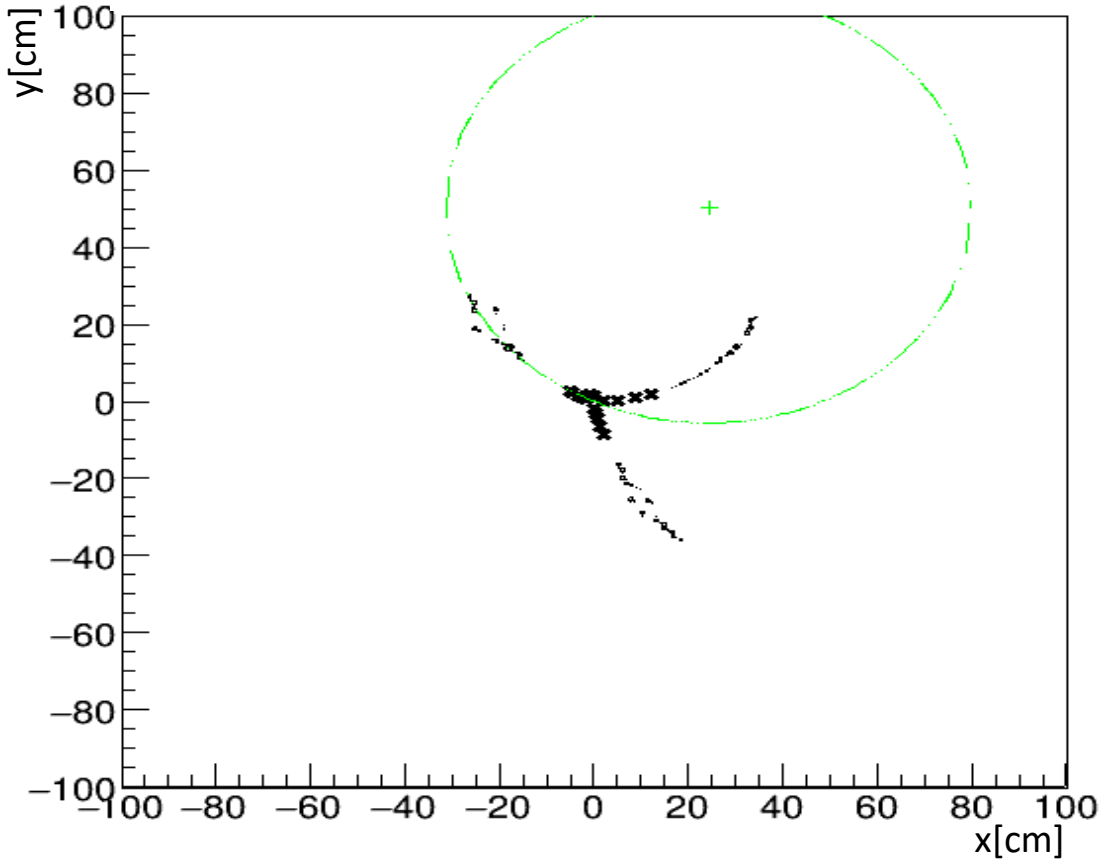
- Different colors for different MC tracks

+ MC track 1
+ MC track 2

SPEED UP MERGE

Trying to find a better handling of the Hough transformation and merging to reduce the runtime:

→ investigated correlations in Hough space



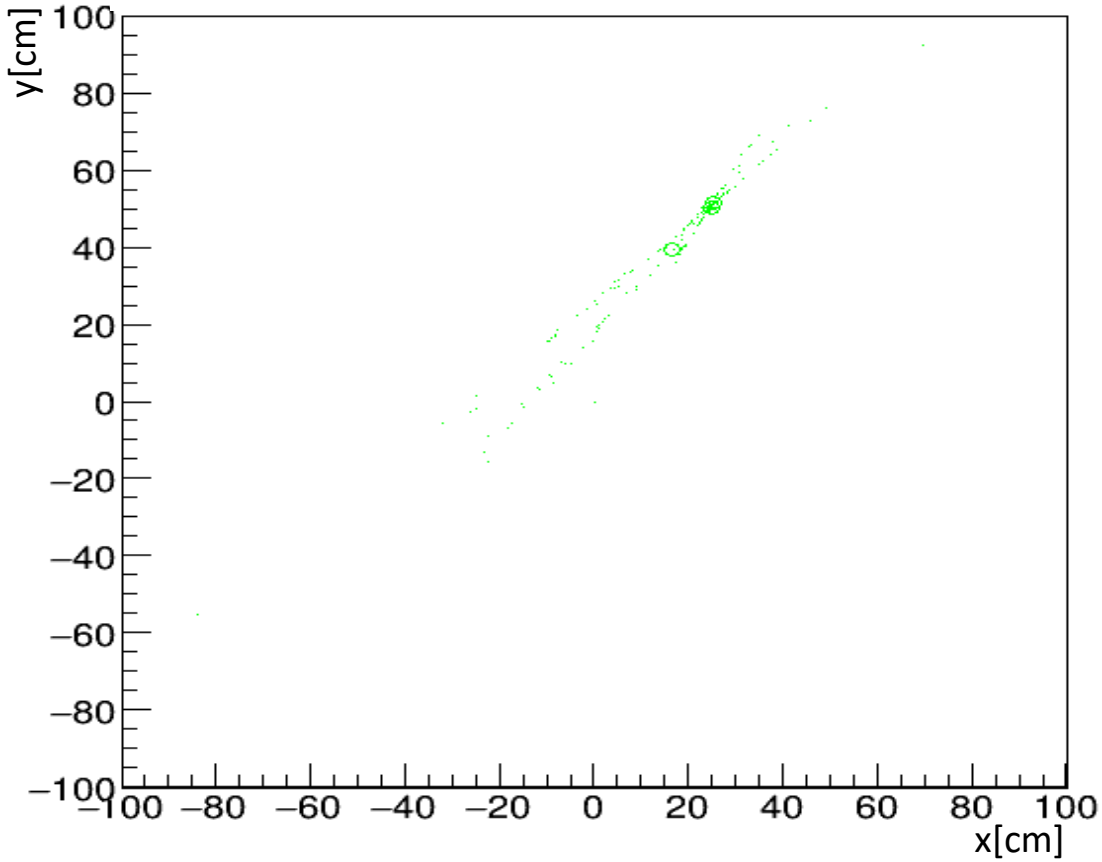
- Different colors for different MC tracks

+ MC track 1
+ MC track 2
+ MC track 3

SPEED UP MERGE

Trying to find a better handling of the Hough transformation and merging to reduce the runtime:

→ investigated correlations in Hough space



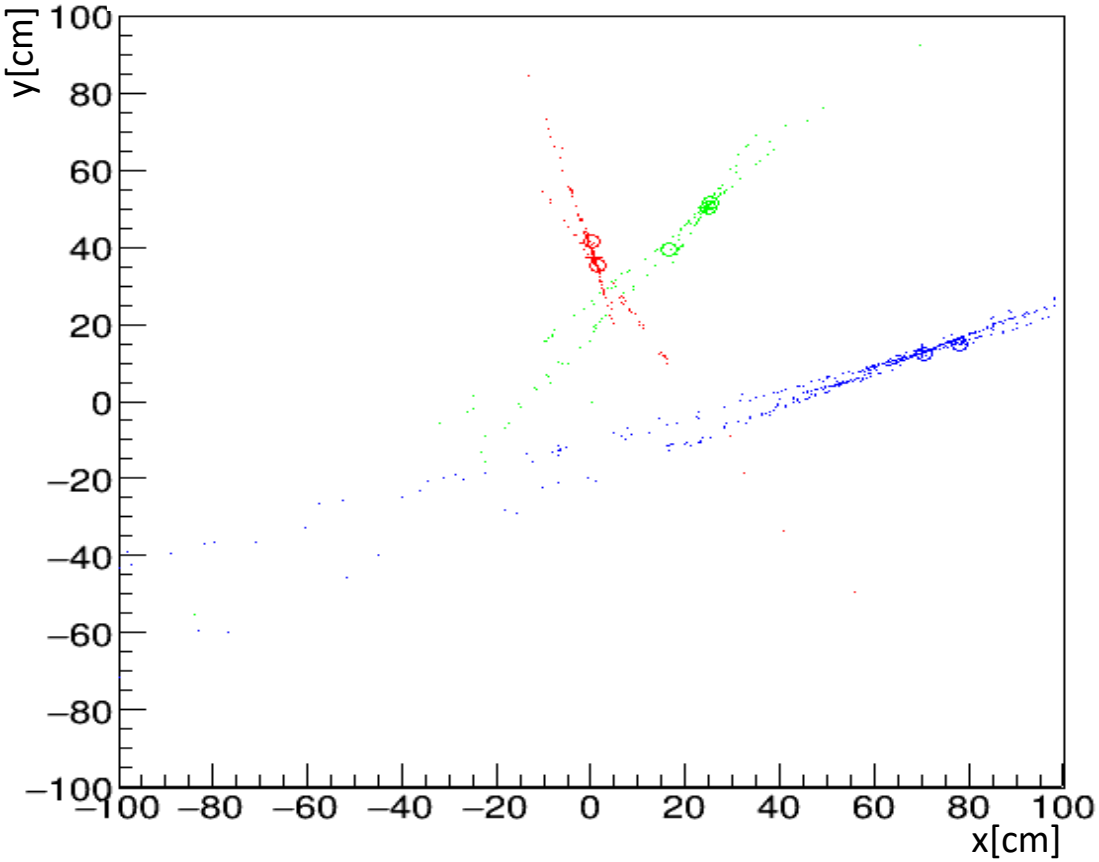
- Different colors for different MC tracks
- Hough space calculated with Apollonius

+ MC track 1
+ MC track 2
+ MC track 3
o Maxima in Hough space for MC track 3

SPEED UP MERGE

Trying to find a better handling of the Hough transformation and merging to reduce the runtime:

→ investigated correlations in Hough space



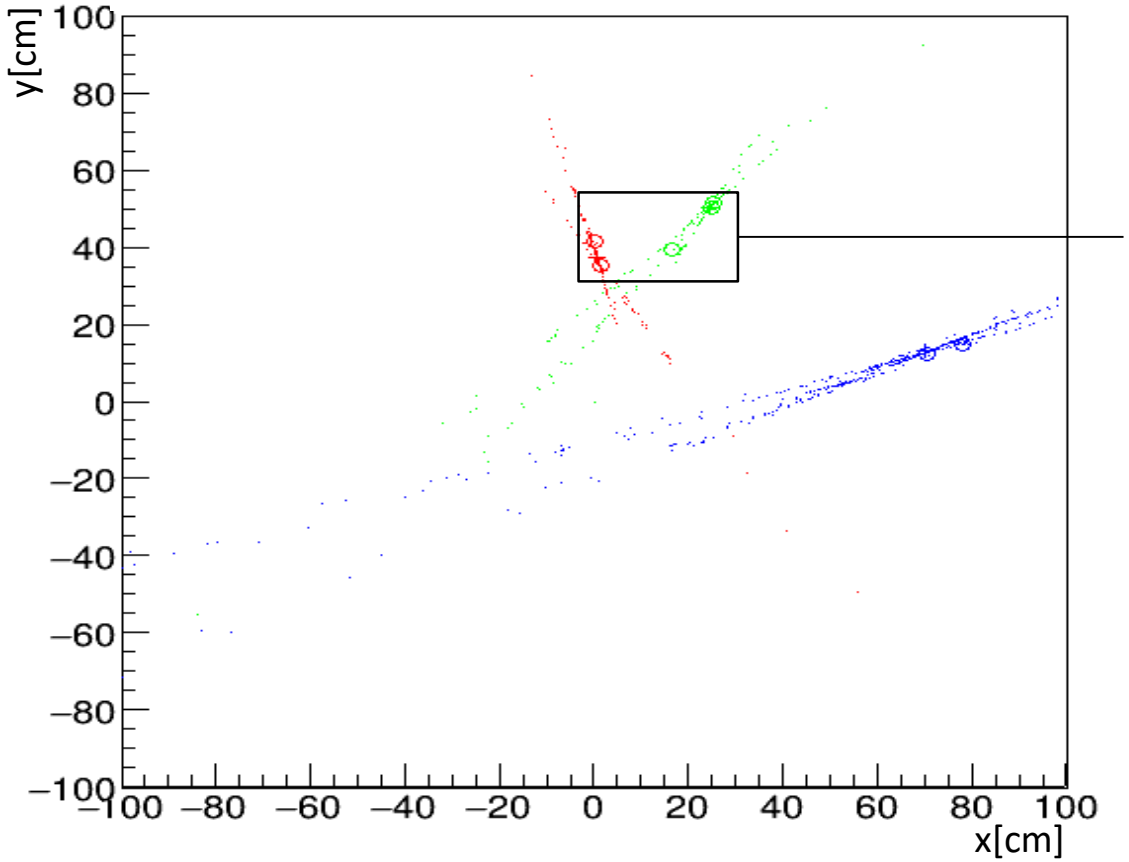
- Different colors for different MC tracks
- Hough space calculated with Apollonius

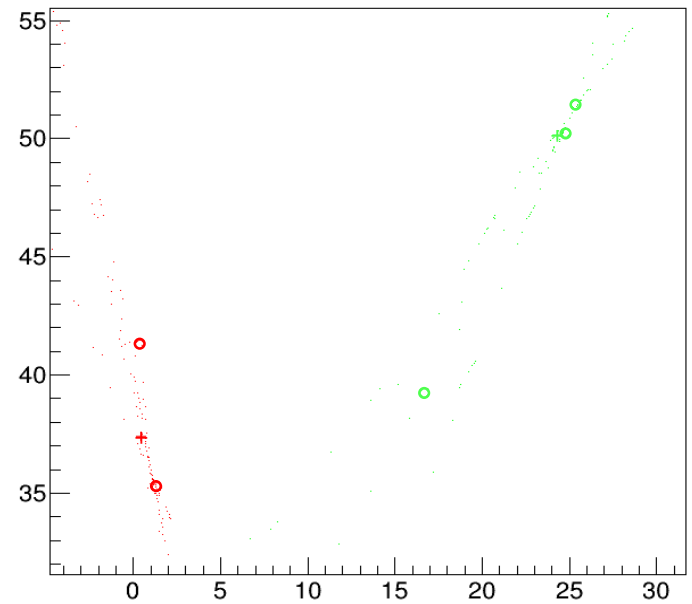
+ MC track 1
+ MC track 2
+ MC track 3
o Maxima for MC track 1
o Maxima for MC track 2
o Maxima for MC track 3

SPEED UP MERGE

Trying to find a better handling of the Hough transformation and merging to reduce the runtime:

→ investigated correlations in Hough space

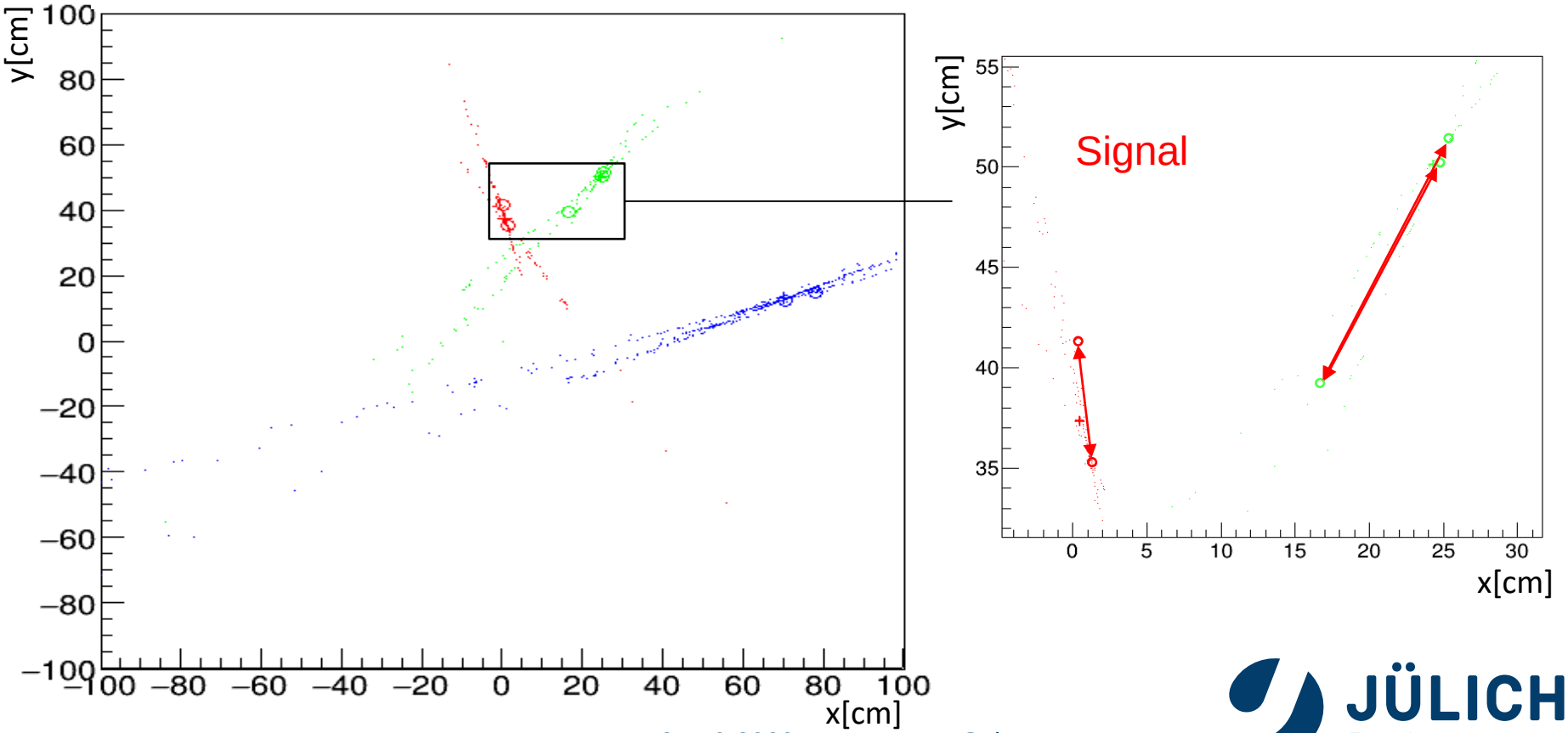




SPEED UP MERGE

Trying to find a better handling of the Hough transformation and merging to reduce the runtime:

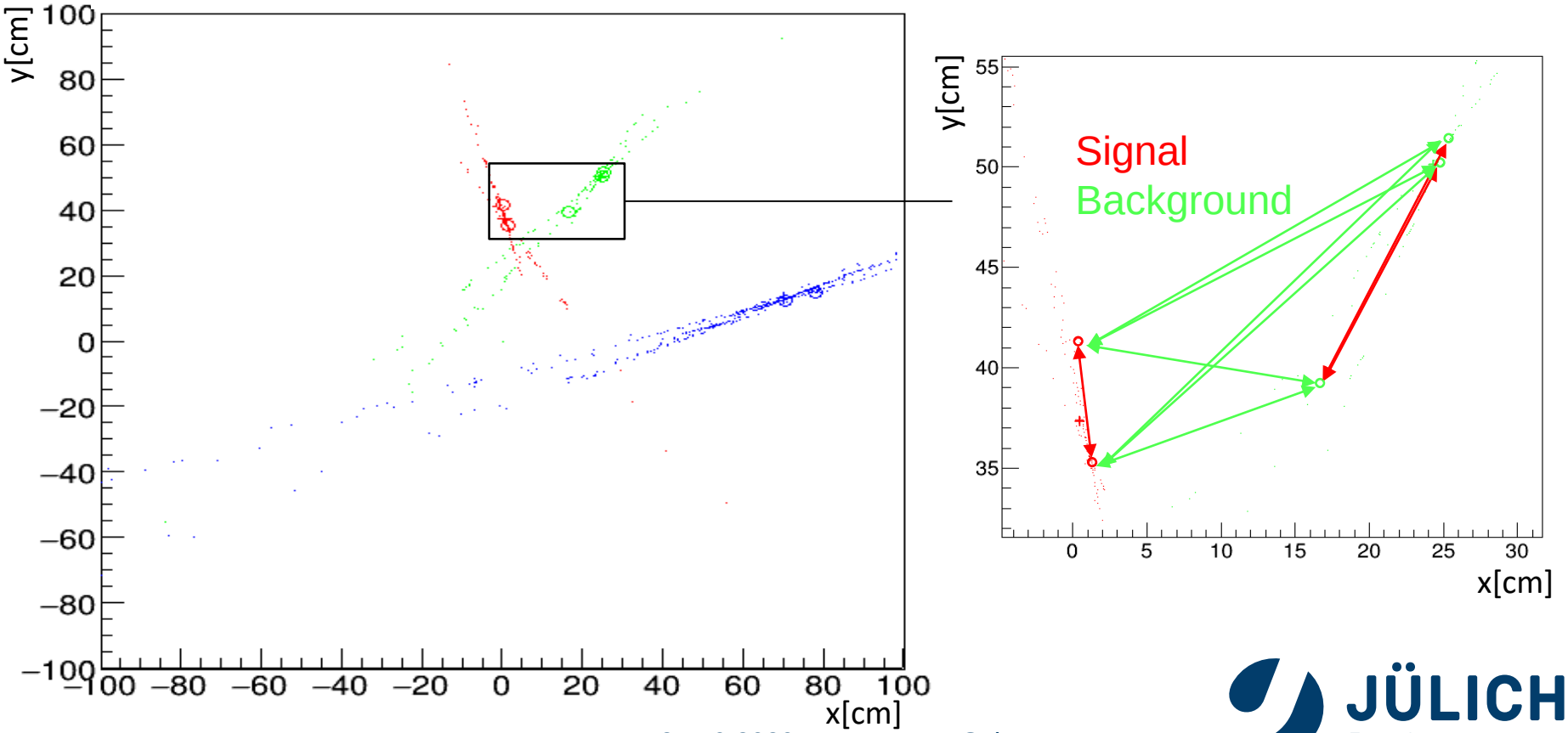
→ investigated correlations in Hough space



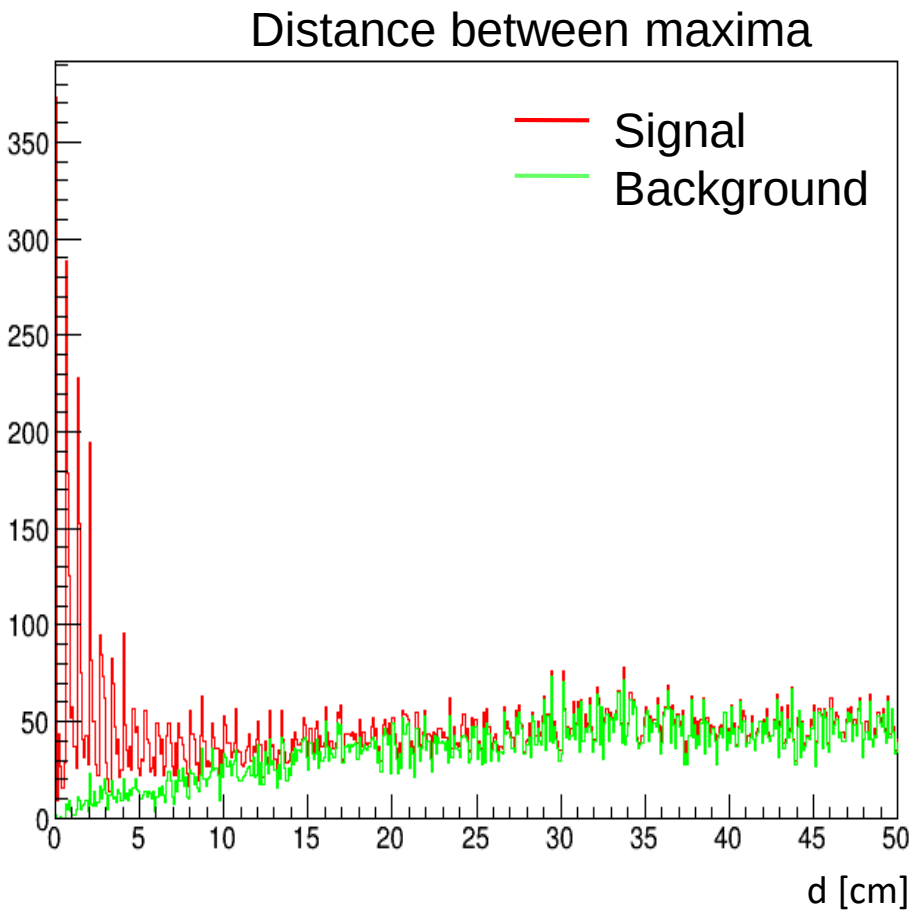
SPEED UP MERGE

Trying to find a better handling of the Hough transformation and merging to reduce the runtime:

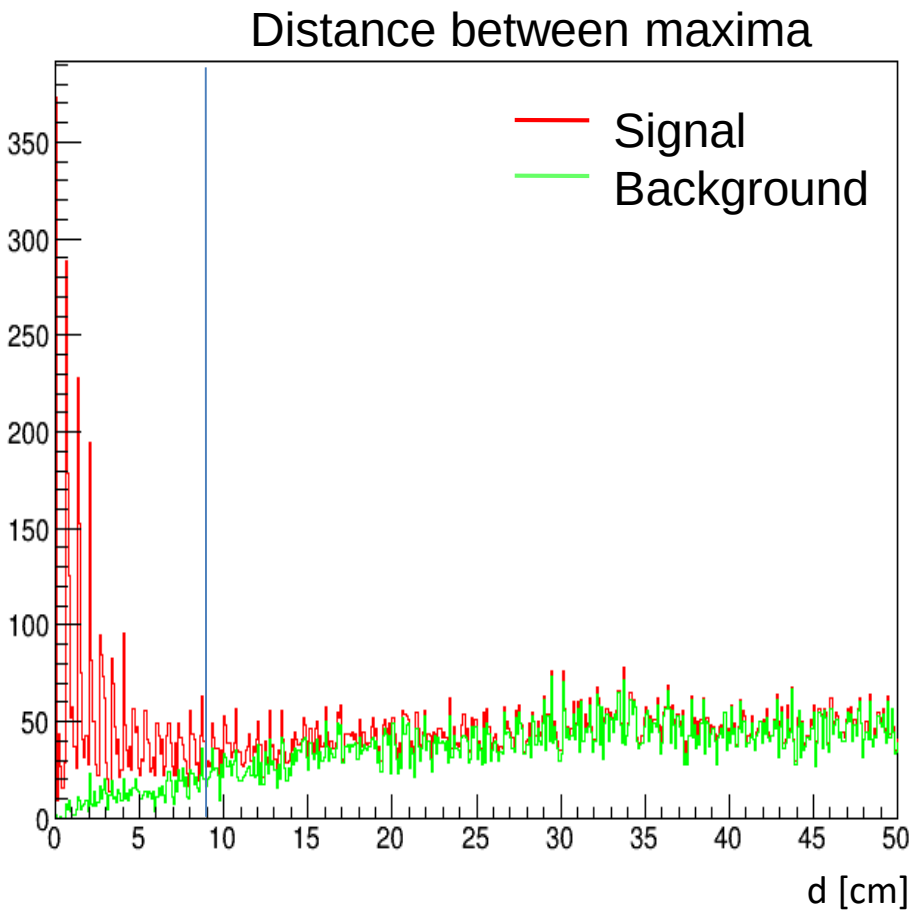
→ investigated correlations in Hough space

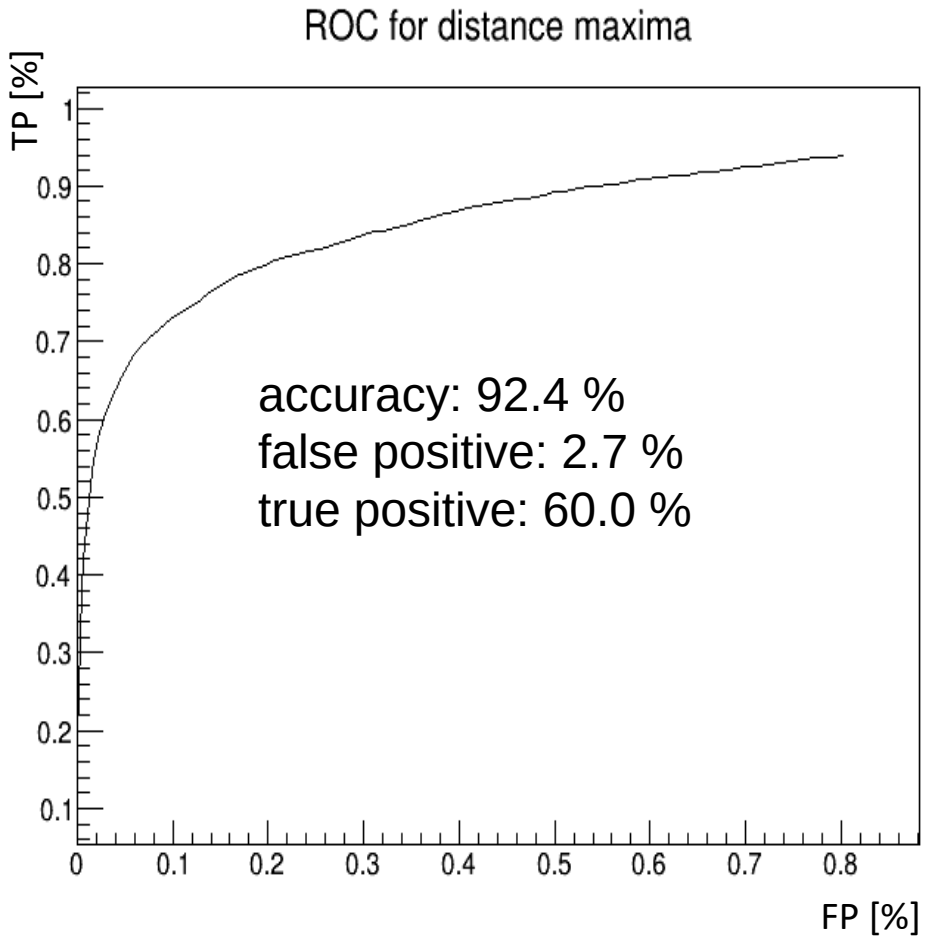


SPEED UP MERGE

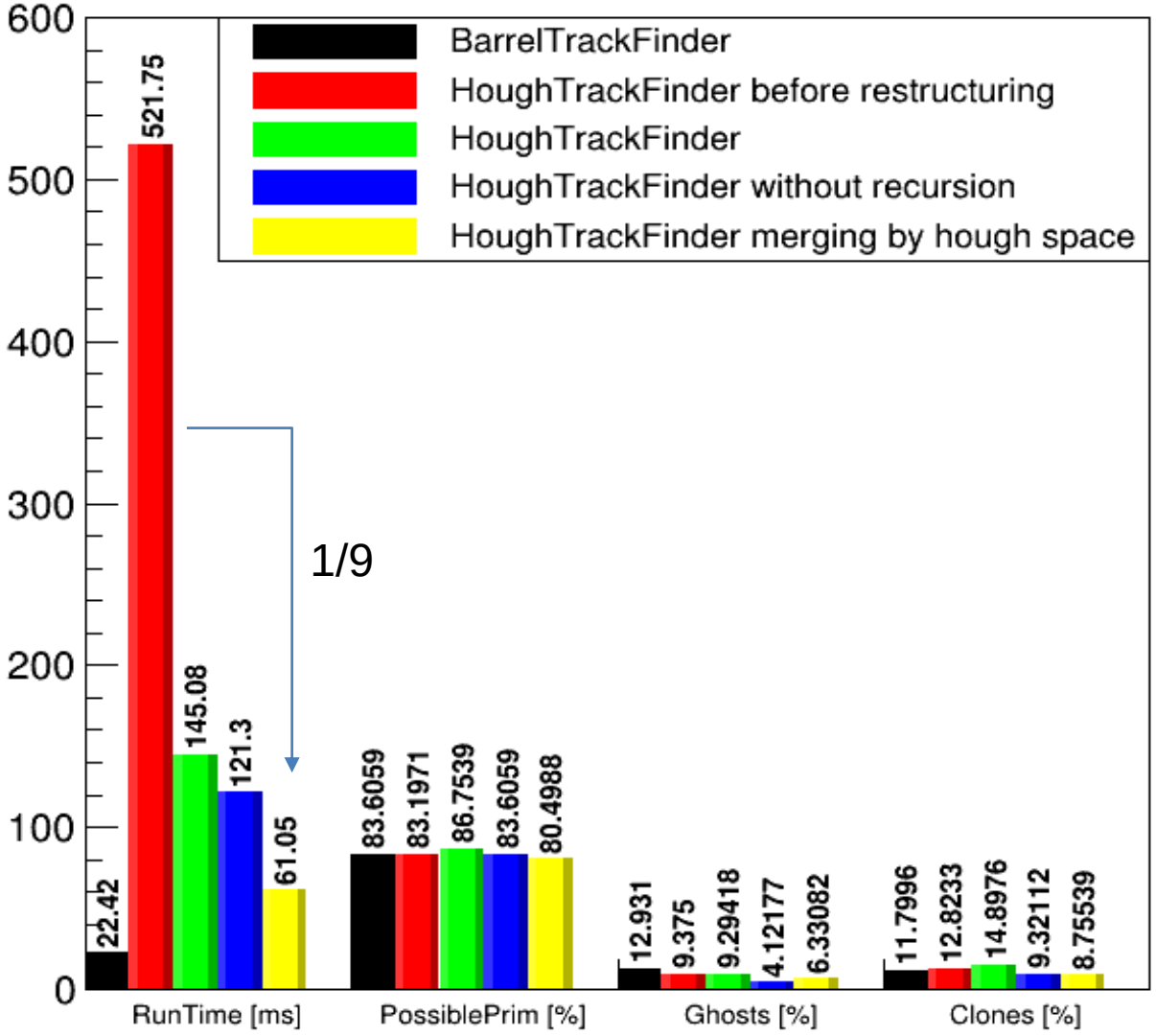


SPEED UP MERGE





SPEED UP MERGE



SUMMARY & OUTLOOK

Summary

- Include HoughTrackFinder into PandaRoot
- Using new data structure and merging method

  ➔ Speed up by a factor of 9

Outlook

- Expand to find secondaries
- Further speed up

Thank you for
your attention!