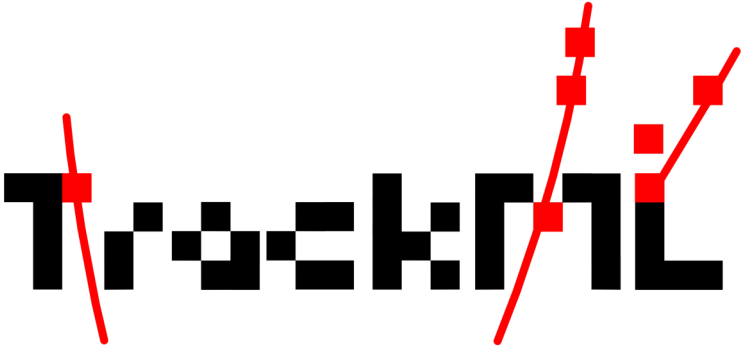
Other solutions submitted to the TrackML challenge

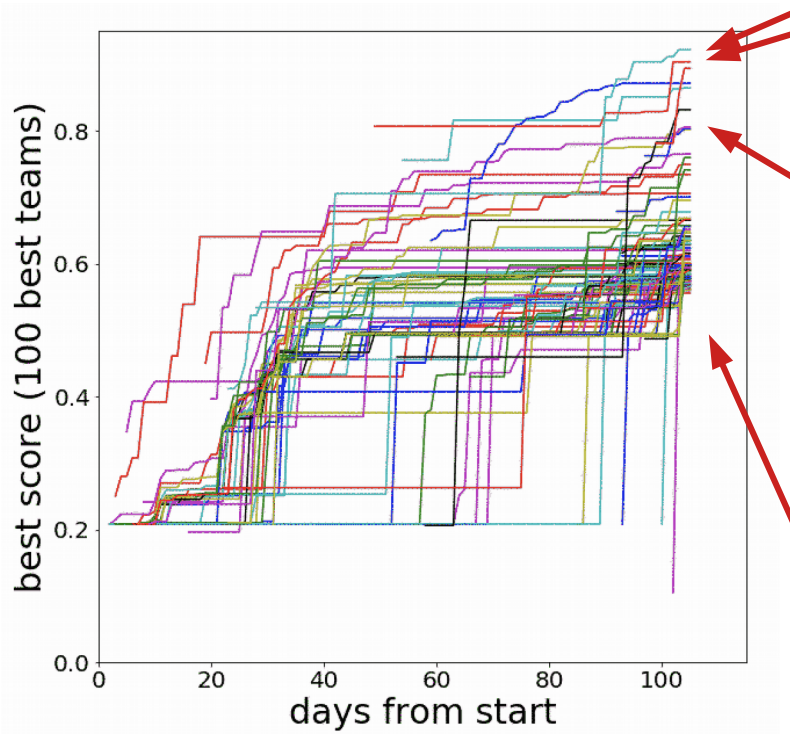


Moritz Kiehn
Université de Genève

For the TrackML organizers and participants

TrackML Grand Finale, CERN, 02.07.2019

Accuracy phase



In the money | Gold | Silver | Bronze

#	△pub	Team Name	Kernel	Team Members	Score	Entries	Last
1	—	Top Quarks			0.92182	10	1y
2	—	outrunner			0.90302	9	10mo
3	—	Sergey Gorbunov	Dedicated talk		0.89353	6	10mo
4	—	demelian	Dedicated talk		0.87079	35	1y
5	—	Edwin Steiner			0.86395	5	10mo
6	—	Komaki			0.83127	22	10mo
7	—	Yuval & Trian	Dedicated talk		0.80414	56	10mo
8	—	bestfitting			0.80341	6	10mo
9	—	DBSCAN forever			0.80114	23	10mo
10	—	Zidmie & KhaVo			0.76320	26	1y
11	—	Andrea Lonza			0.75845	15	10mo
12	—	Finnies	Dedicated talk		0.74827	56	10mo
13	—	Rei Matsuzaki			0.74035	12	10mo
14	—	Mickey			0.73217	10	1y
15	—	Vicens Gaitan			0.70429	19	1y
16	—	Robert			0.69955	3	1y
17	—	Yuval-CPMP tribute band			0.69364	20	1y
18	—	N. Hi. Bouzu			0.67573	9	1y
19	—	Steins;Gate			0.66763	12	1y
20	▲1	Victor Nedel'ko			0.66723	4	1y
21	▼1	atom1231 & Kent AI Lab			0.66320	42	10mo
22	▲1	Nerdiholic			0.65420	12	1y
23	▼1	Sergey Zlobin			0.65352	23	1y

100

Accuracy #9: DBSCAN forever (Jury Clustering Prize)

Jean-Francois Puget “CPMP”

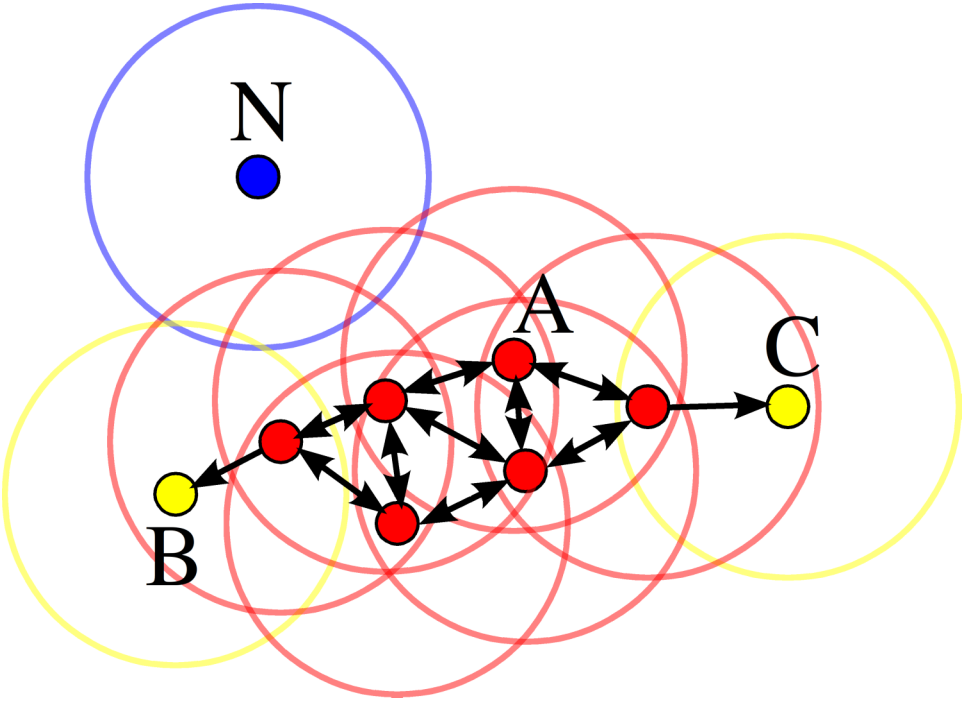
Software engineer at IBM in France

https://github.com/jfpuget/Kaggle_TrackML

DBSCAN?

Density-based clustering

- Few parameters:
  distance, min #, (metric)
- Simple and available
- Used in starting kit
  score ≈ 0.2



wikipedia.org/wiki/DBSCAN

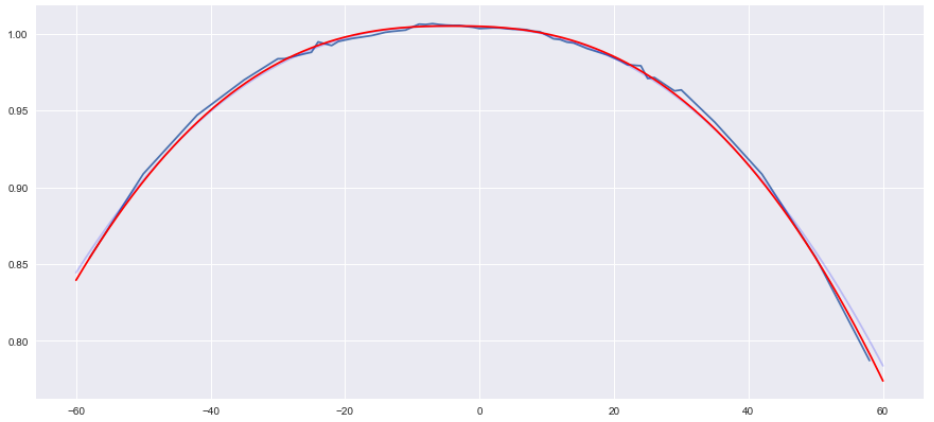
DBSCAN forever – Improvements

Hough-transform-like unfolding for helix model

- Pick a (r_0, z_0) pair
- Compute ρ, ϕ, η-like for each hit
- Assumes d0 = 0

Run for many (r_0, z_0) pairs

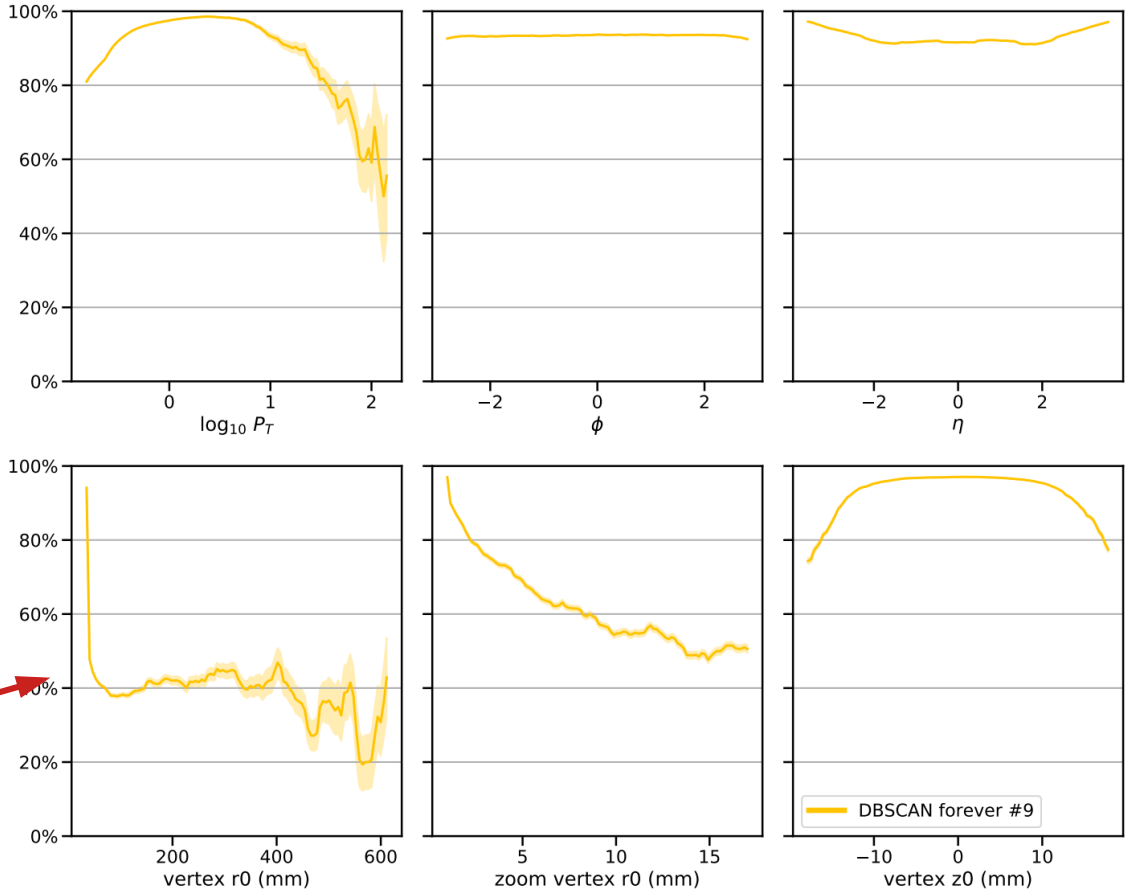
Different parameters for inner/ outer detectors



Magnetic field extracted from data

From CPMP Kaggle post

DBSCAN forever – Efficiencies



Probably:

$d_0 = 0$ assumption in helix unrolling

DBSCAN forever – Take away

Manually tuned, classical algorithm with smart preprocessing

Implementation

- Pure python
- DBSCAN from scikit-learn

Runtime

- 3Gb per worker
- Timing unknown

Accuracy #2: outrunner

Pei-Lien Chou

Software engineer image-based deep learning in Taiwan.

Kaggle Notebook

outrunner – Setup

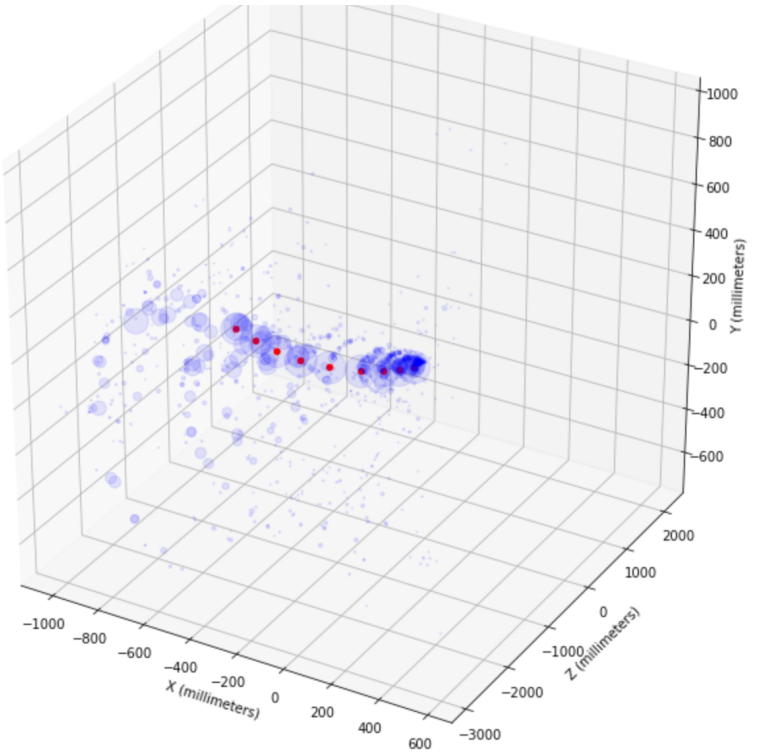
Train DNN on hit pairs

- 27 inputs (x,y,z,cells,...)
- 4k-2k-2k-2k-1k hidden layers

Compute full hit adjacency matrix: probability P(i,j) that 2 hits match

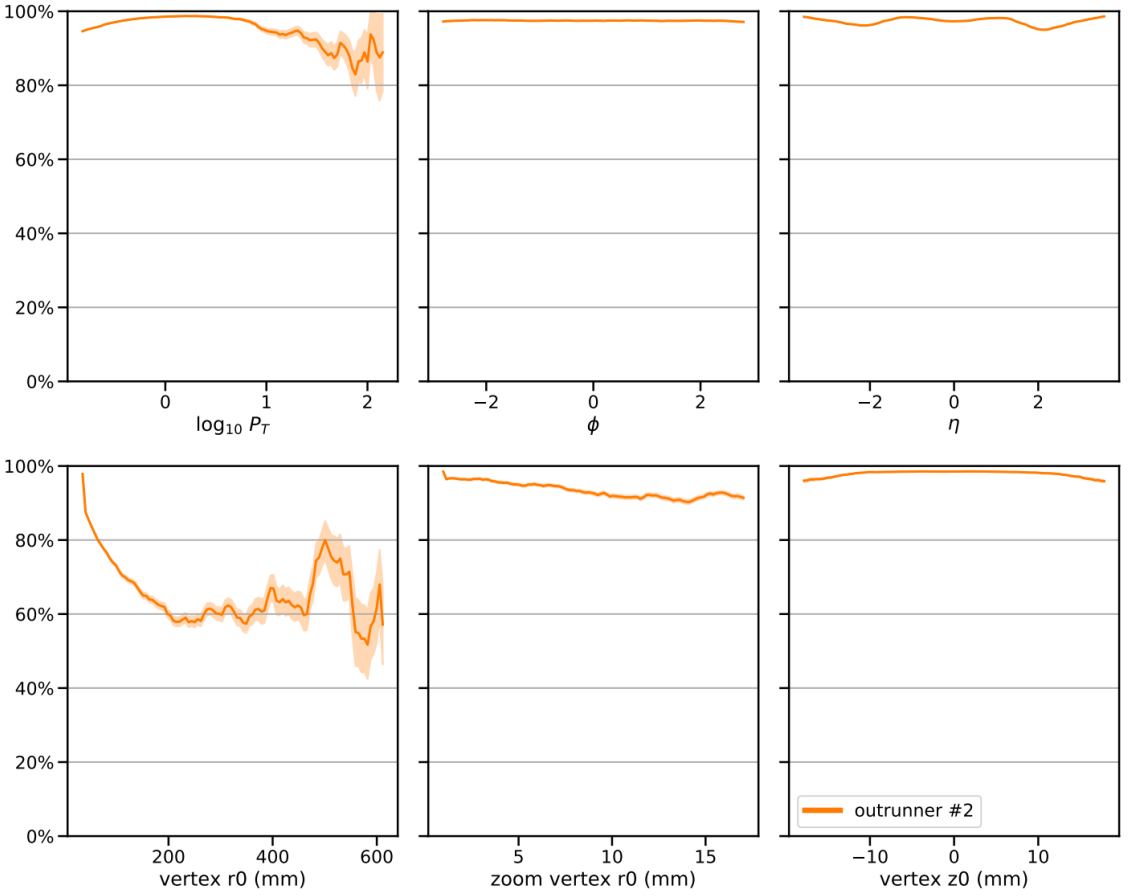
Pick high probability comb.

Helix-like fit for cleaning



Graphics from outrunner

outruner – Efficiencies



outrunner – Take away

True Deep Learning Solution

- No track following
- No geometric modelling

But: slow execution

Implementation

- Pure python
- Keras for ML

Runtime

- multiple hours / event

Accuracy #1: Top Quarks

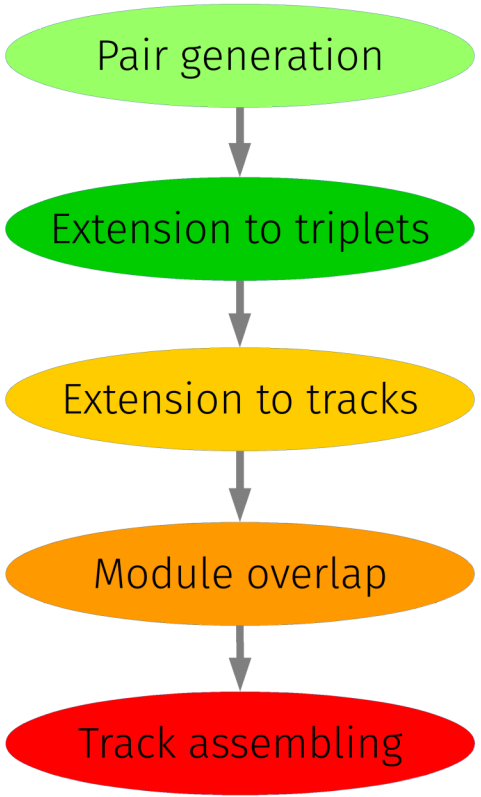
Johan Sokrates Wind “icecuber”

Industrial Mathematics Master student in Norway (main contributor)

Erling Solberg “erlinsol”

https://github.com/top-quarks/top-quarks

Top Quarks – Overview



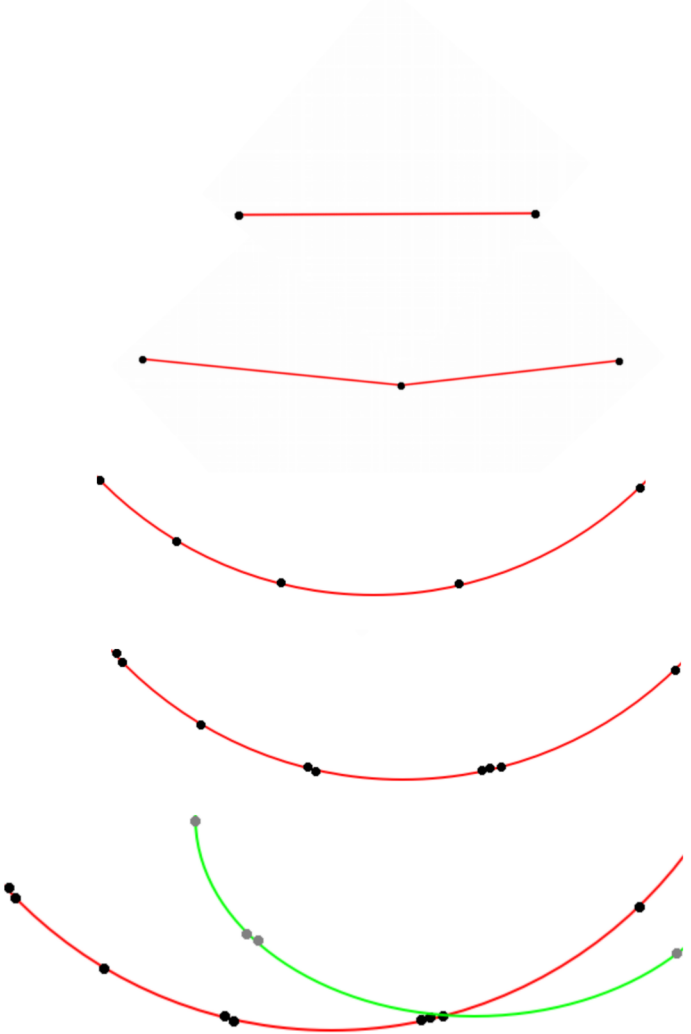


Illustration from J-R. Vlimant

Illustration from J.S. Wind

Top Quarks – Pair generation

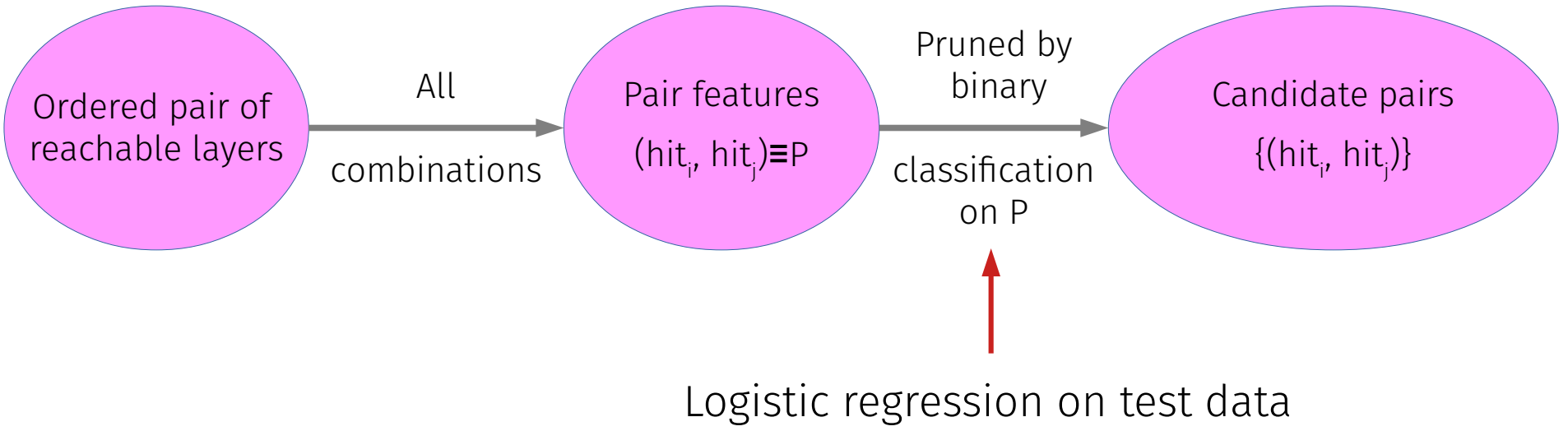


Illustration from J-R. Vlimant

Top Quarks – Extension to triplets

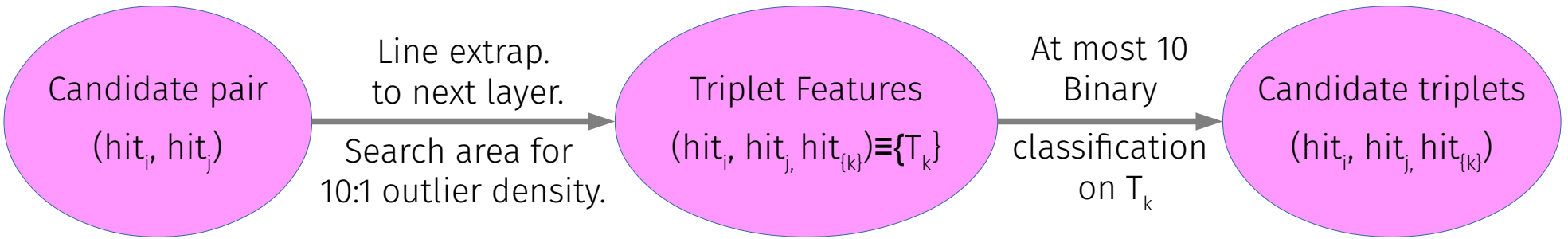
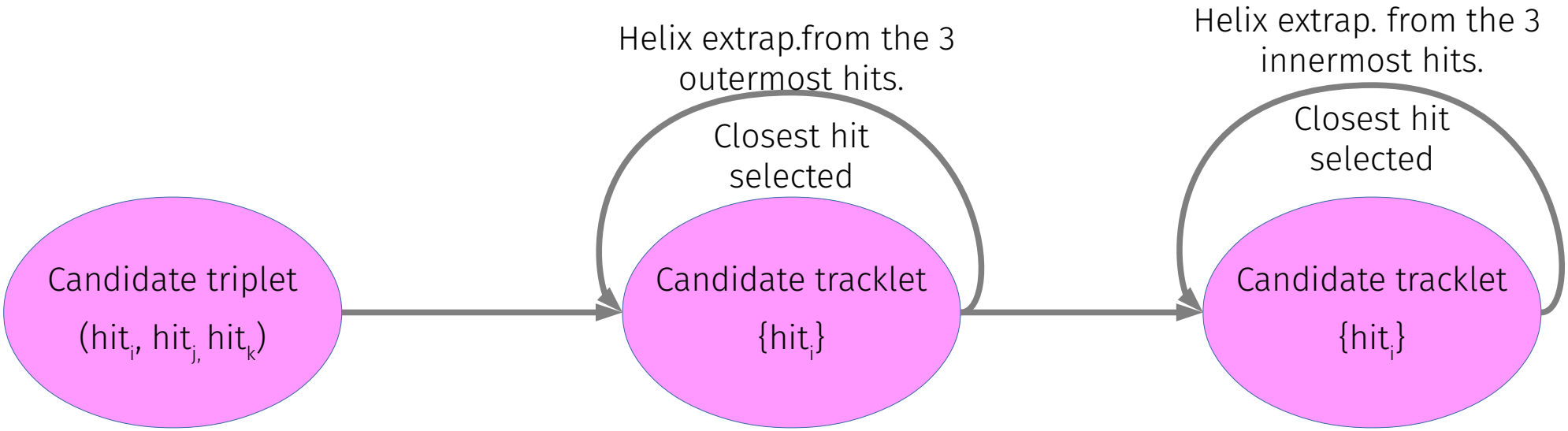


Illustration from J-R. Vlimant

Top Quarks – Extension to tracklets



Extrapolation w/ 2nd order circle approximation

Magnetic field from data

Illustration from J-R. Vlimant

Top Quarks – Module overlap

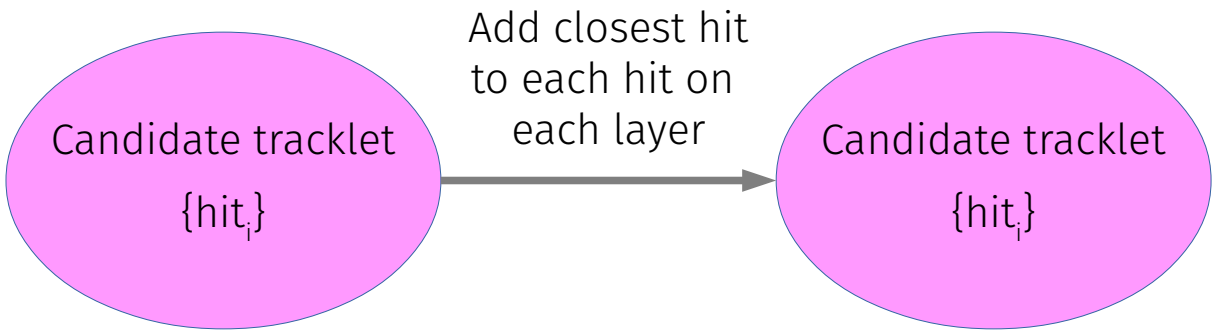
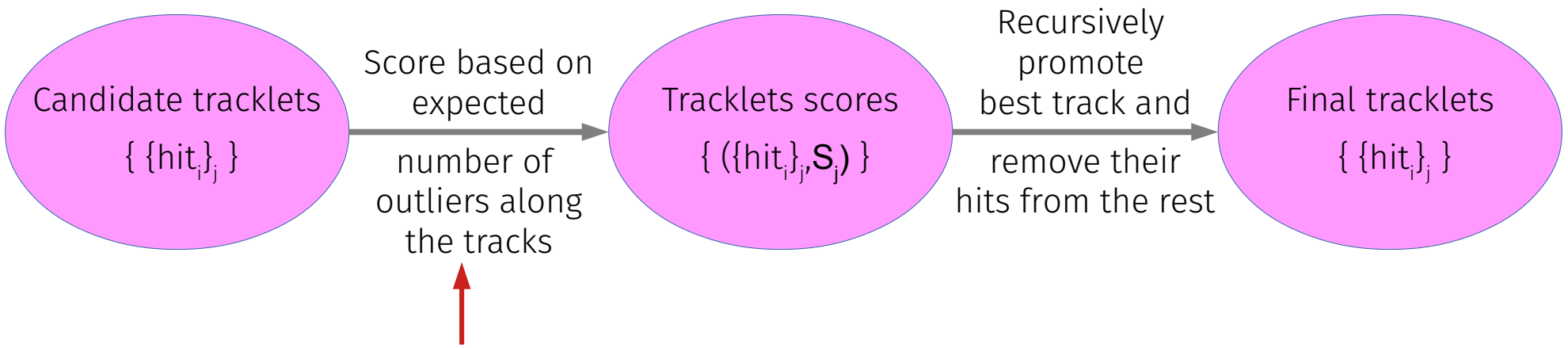


Illustration from J-R. Vlimant

Top Quarks – Track assembly



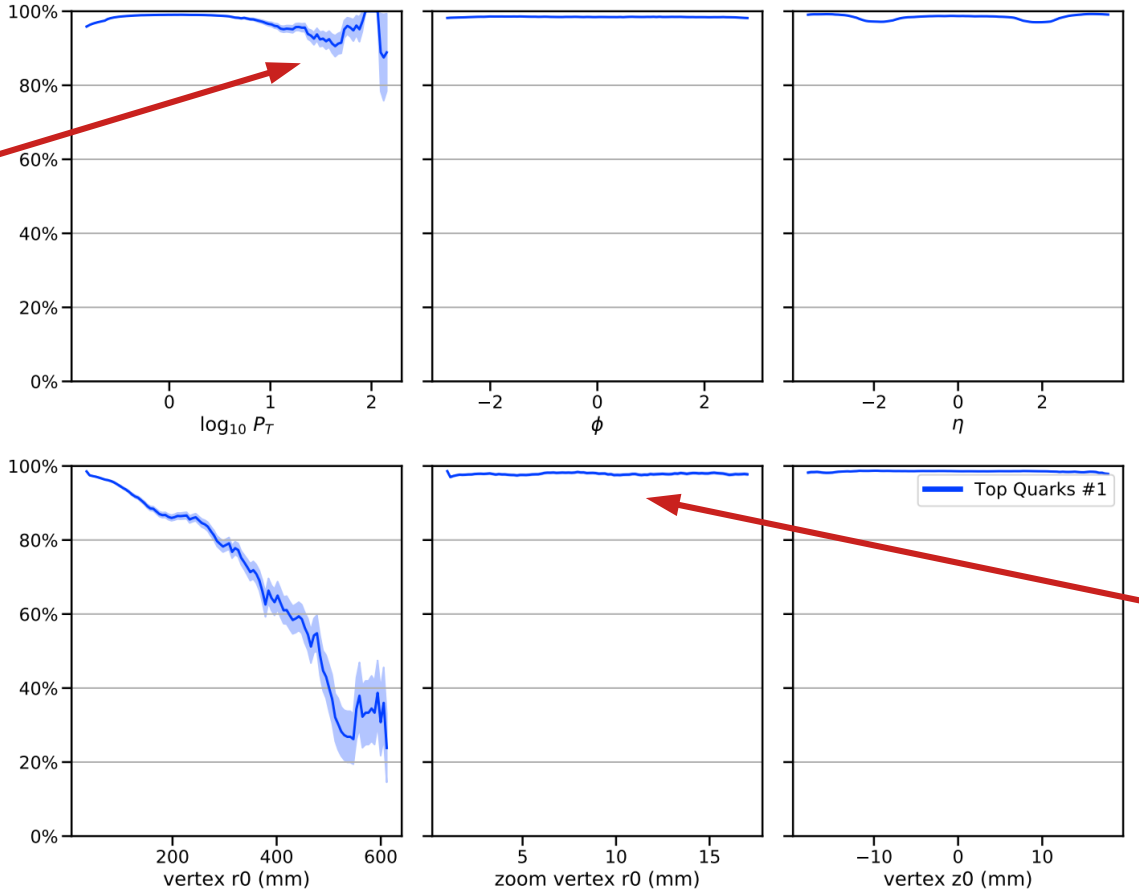
Interesting idea:

Model noise instead of signal

Illustration from J-R. Vlimant

Top Quarks – Efficiencies

A bit strange, but exists in almost every submission



Good

Top Quarks – Take away

Custom algorithm:

Track following with ML sprinkles on top

Custom implementation w/ fast runtime enables fast experimentation

Served as inspiration for throughput phase, e.g. #3 Marcel Kunze

Implementation

- Custom C++ code
- Custom quad-tree based hit lookup
- Python/scikit-learn for training

Runtime

- 8min / event
- Memory 2.8Gb avg, 4Gb max

Accuracy #100: diogo (Organizer's pick)

Diogo R. Ferreira

Researcher at the University of Lisbon,
focusing on data science and nuclear fusion

https://github.com/diogoff/trackml-100

diogo – Routes

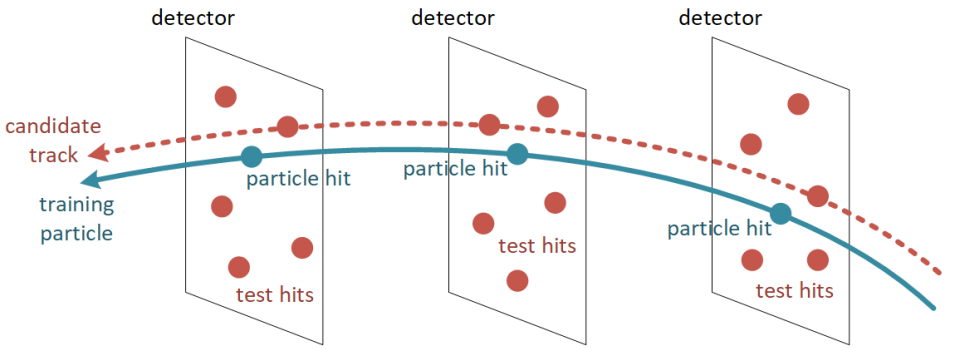
Build routes from truth

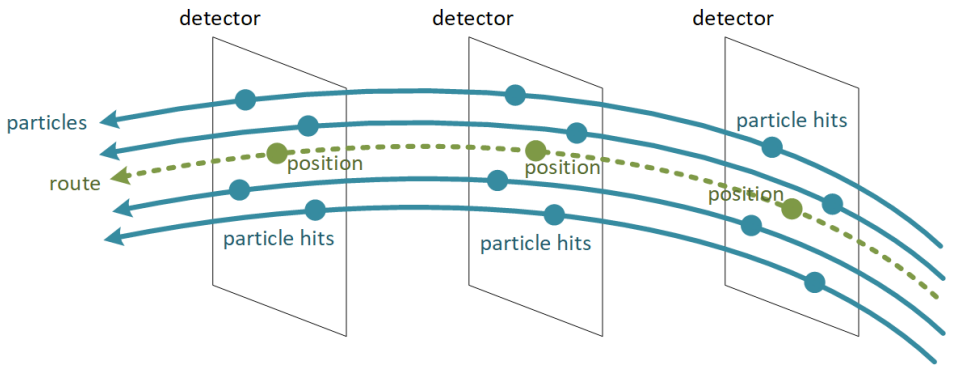
- All seen sequences of traversed modules
- Average estimates for shared sequences

On reconstruction

- Pick closest route(s) to hit
- Select route by distance

Similar to LHC triggers

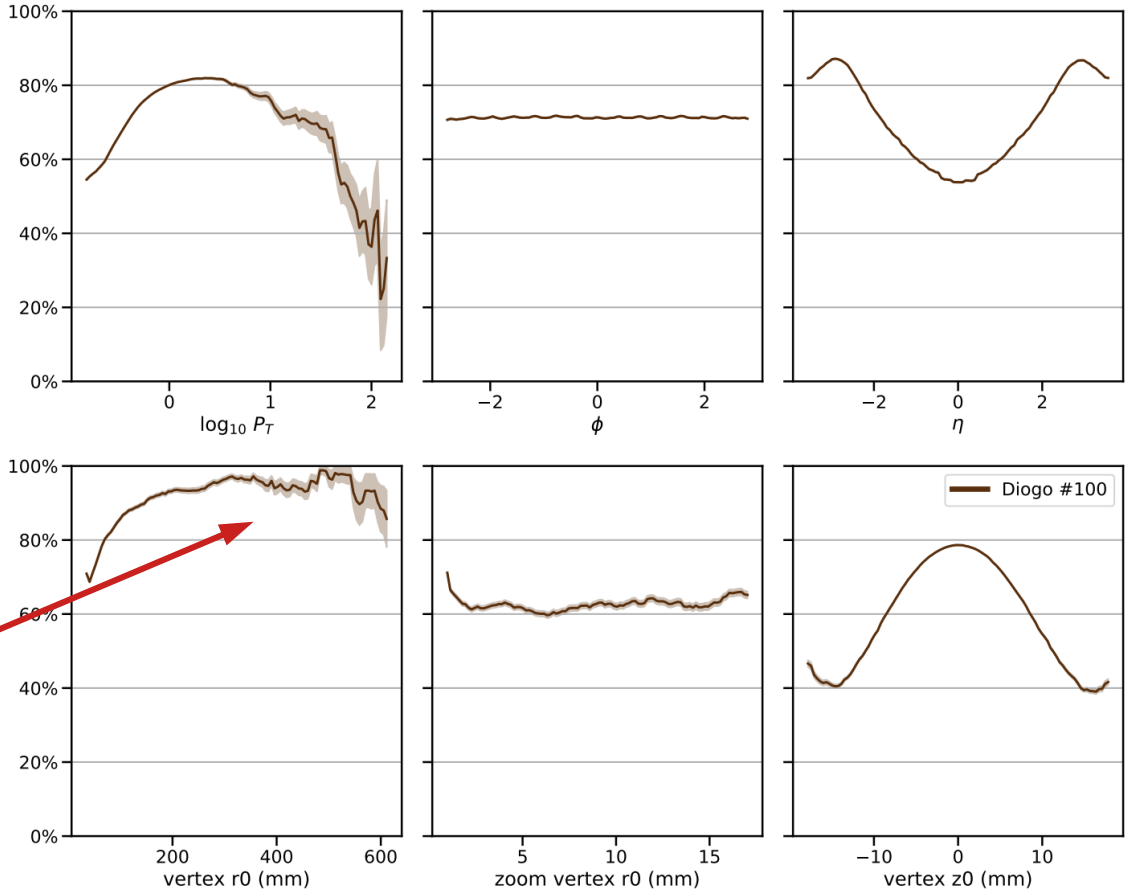




Graphics from github.com/diogoff/trackml-100

diogo – Efficiencies

High performance for displaced vertices

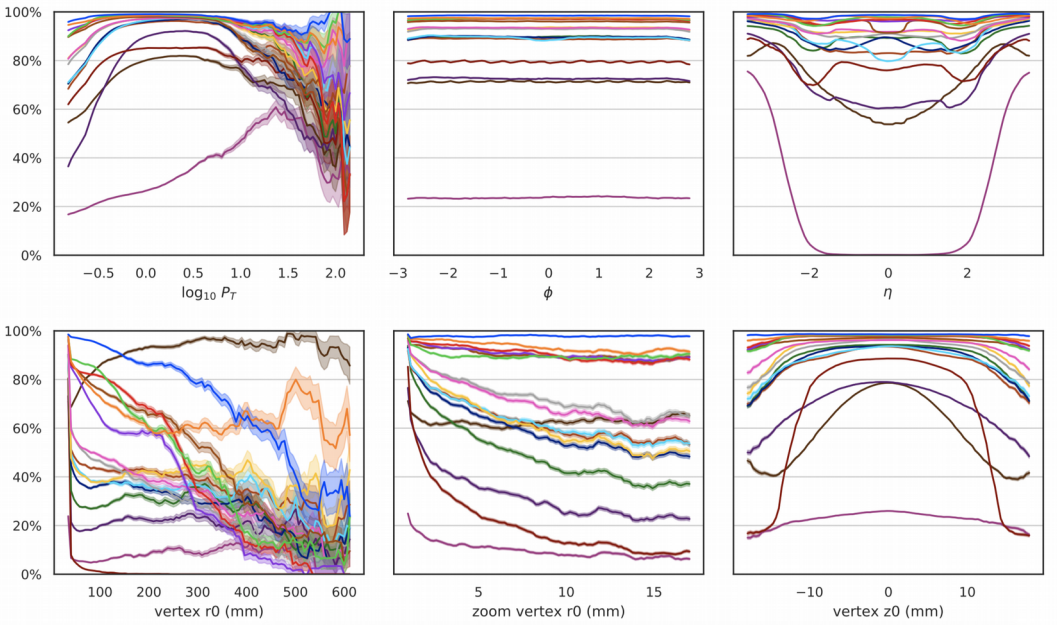


Summary

Interesting solutions from non-domain experts

Simple algorithms can be quite powerful

But, this is a complex problem that sometimes requires complex solutions



Details e.g. in NEURIPS chapter
arXiv:1904.06778