Machine Learning for Primary Vertex Reconstruction

Rui Fang[1] Henry Schreiner[1, 2] Mike Sokoloff[1] Constantin Weisser[3] Mike Williams[3]

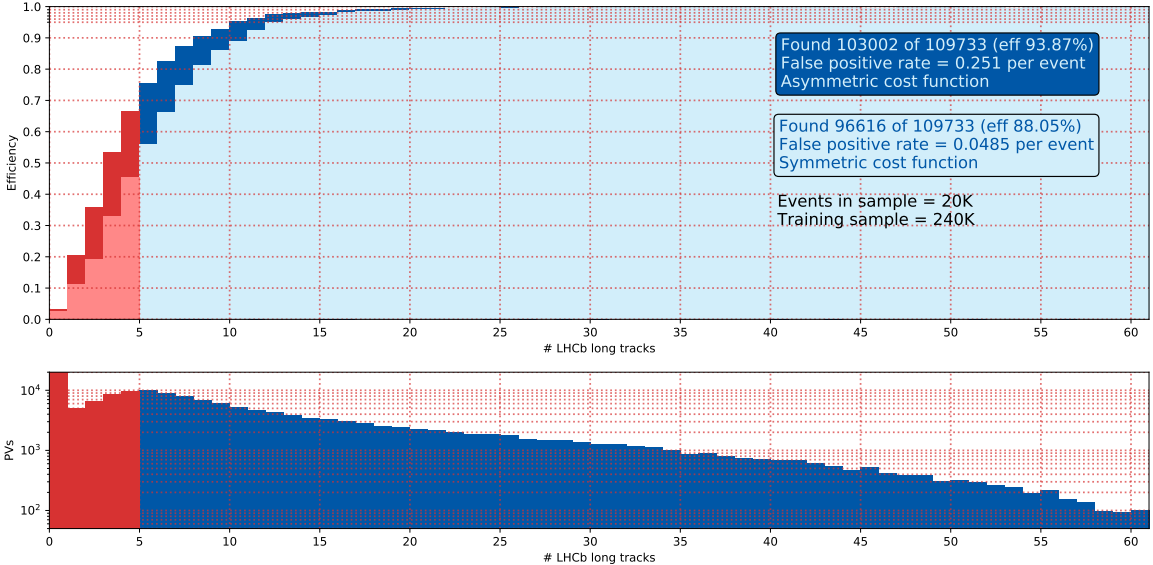
March 20, 2019

[1]The University of Cincinnati

[2]Princeton University

[3]Massachusetts Institute of Technology

Supported by:

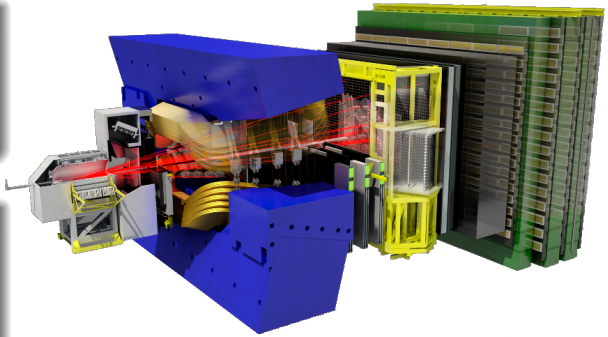

Found 103002 of 109733 (eff 93.87%)
False positive rate = 0.251 per event
Asymmetric cost function
Found 96616 of 109733 (eff 88.05%)
False positive rate = 0.0485 per event
Symmetric cost function
Events in sample = 20K
Training sample = 240K
Efficiency
LHCb long tracks
PVs
LHCb long tracks

The changes

- 30 MHz software trigger
- 7.6 PVs per event (Poisson distribution)
- Roughly 5.5 visible PVs per event

The problem

- Much higher pileup
- Very little time to do the tracking
- Current algorithms too slow

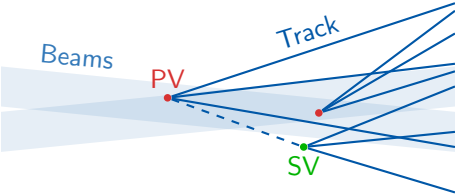


We need to rethink our algorithms from the ground up...

Vertices and tracks

Vertices

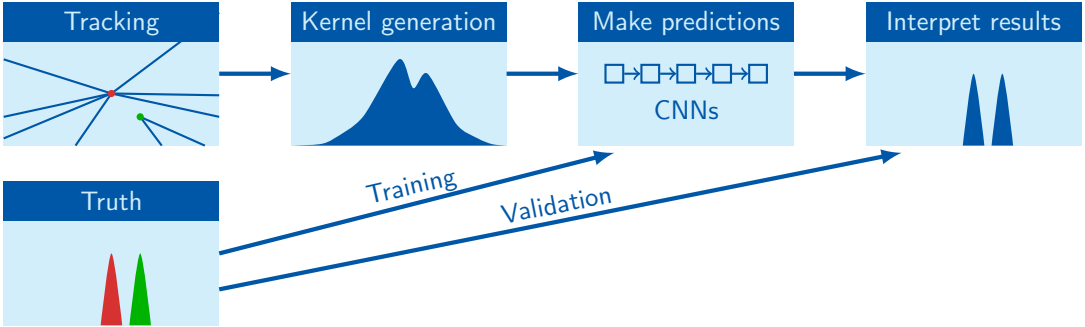
- Events contain ≈ 7 Primary Vertices (≈ 5 visible PVs)
  - A PV should contain 5+ long tracks
- Multiple Secondary Vertices (SVs) per event as well
  - A SV should contain 2+ tracks



Adapt to machine learning?

- Sparse 3D data (41M pixels) $\rightarrow$ rich 1D data
- 1D convolutional neural nets
- Highly parallelizable, GPU friendly
- Opportunities to visualize learning process

A hybrid ML approach

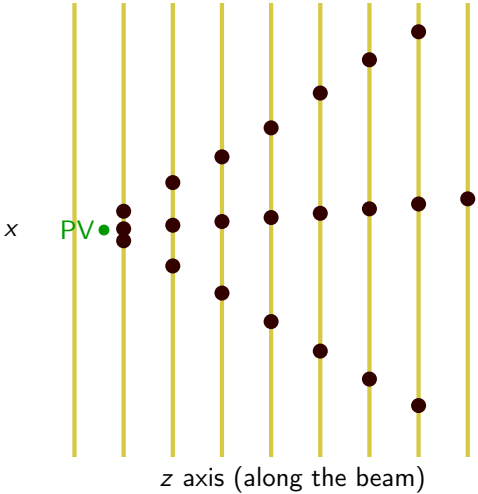


Machine learning features (so far)

- Prototracking converts sparse 3D dataset to feature-rich 1D dataset
- Easy and effective visualization due to 1D nature
- Even simple networks can provide interesting results

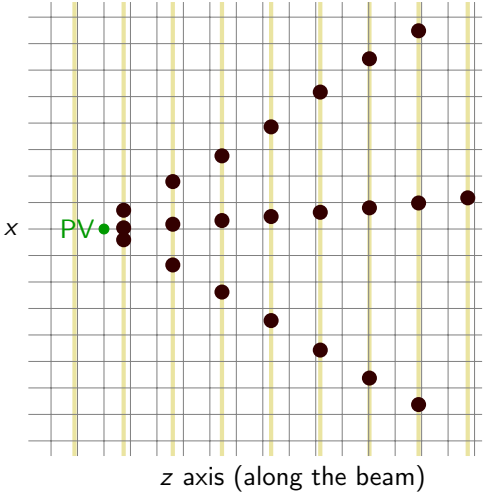
Tracking procedure

- Hits lie on the 26 planes
- For simplicity, only 3 tracks shown



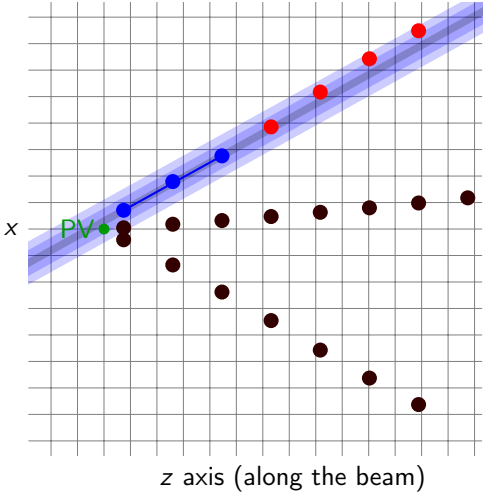
Tracking procedure

- Hits lie on the 26 planes
- For simplicity, only 3 tracks shown
- Make a 3D grid of voxels (2D shown)
- Note: only z will be fully calculated and stored



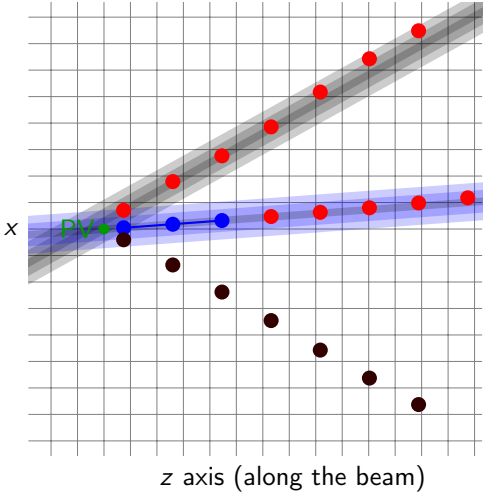
Tracking procedure

- Hits lie on the 26 planes
- For simplicity, only 3 tracks shown
- Make a 3D grid of voxels (2D shown)
- Note: only z will be fully calculated and stored
- Tracking (full or partial)



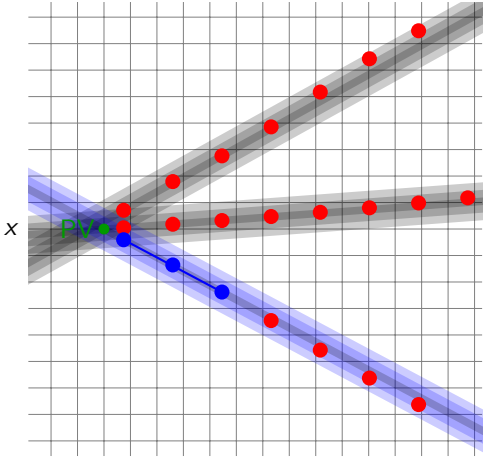
Tracking procedure

- Hits lie on the 26 planes
- For simplicity, only 3 tracks shown
- Make a 3D grid of voxels (2D shown)
- Note: only z will be fully calculated and stored
- Tracking (full or partial)
- Fill in each voxel center with Gaussian PDF



Tracking procedure

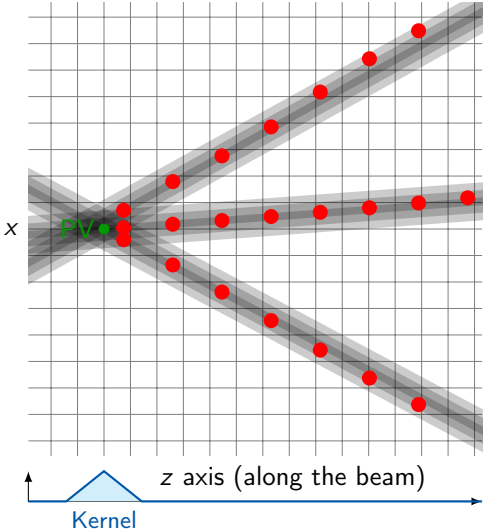
- Hits lie on the 26 planes
- For simplicity, only 3 tracks shown
- Make a 3D grid of voxels (2D shown)
- Note: only z will be fully calculated and stored
- Tracking (full or partial)
- Fill in each voxel center with Gaussian PDF
- PDF for each (proto)track is combined



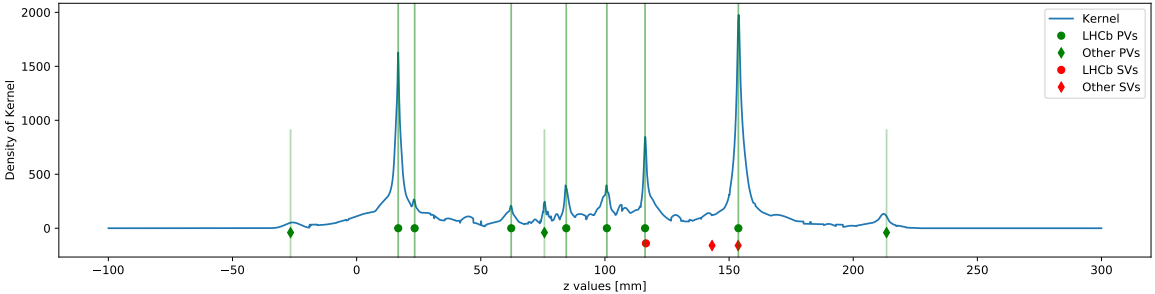
z axis (along the beam)

Tracking procedure

- Hits lie on the 26 planes
- For simplicity, only 3 tracks shown
- Make a 3D grid of voxels (2D shown)
- Note: only z will be fully calculated and stored
- Tracking (full or partial)
- Fill in each voxel center with Gaussian PDF
- PDF for each (proto)track is combined
- Fill z "histogram" with maximum KDE value in xy



Example of z KDE histogram



Note: All events from toy detector simulation

Human learning

- Peaks generally correspond to PVs and SVs

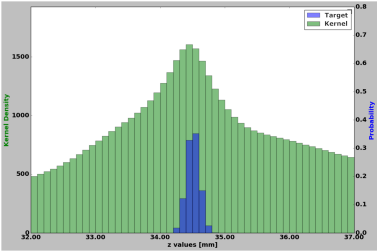
Challenges

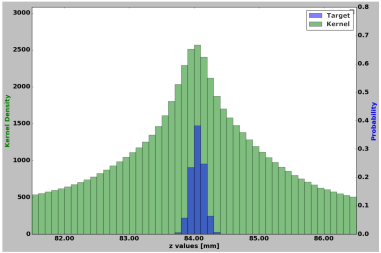
- Vertex may be offset from peak
- Vertices interact

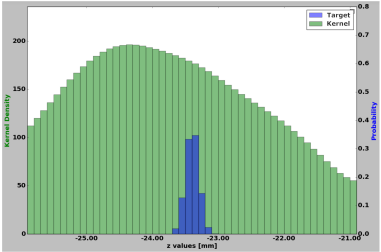
Target distribution

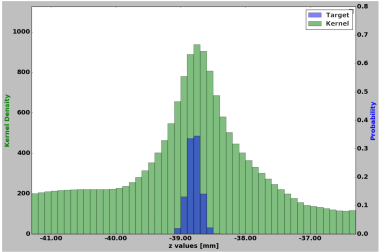
Build target distribution

- True PV position as the mean of Gaussian
- σ (standard deviation) is 100 $\mu\mathrm{m}$ (simplification)
- Fill bins with integrated PDF within ±3 bins (±300 $\mu\mathrm{m}$)

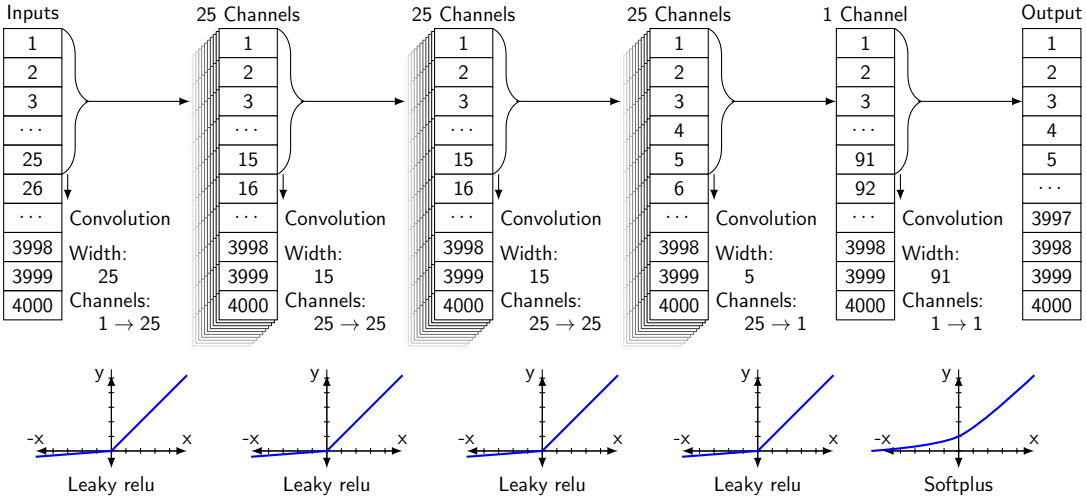




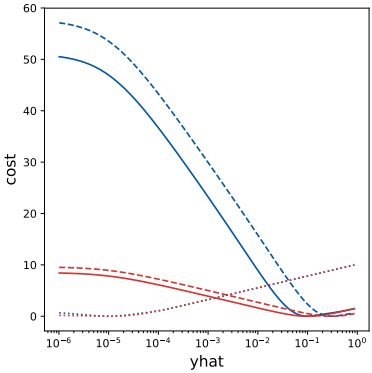


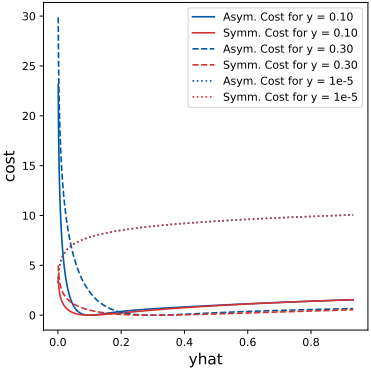


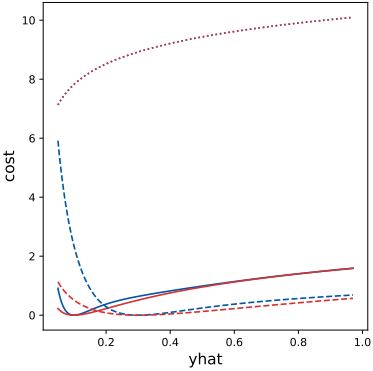
Neural network architecture



Cost function



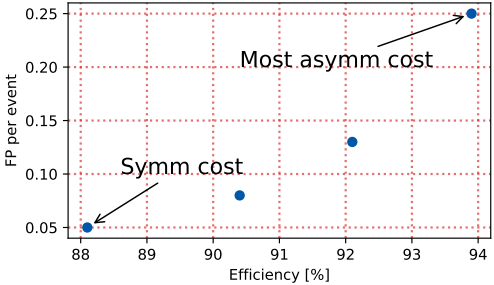


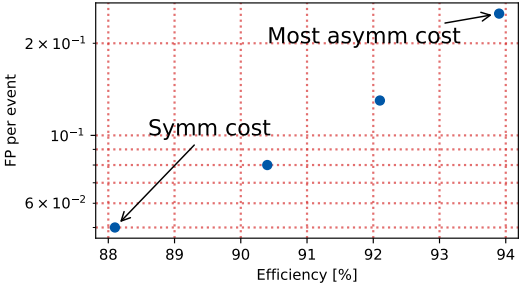


Approach

- Symmetric cost function: low FP but low efficiency
- Adding asymmetry term controls trade-off for FP vs. efficiency

False Positive and efficiency rates





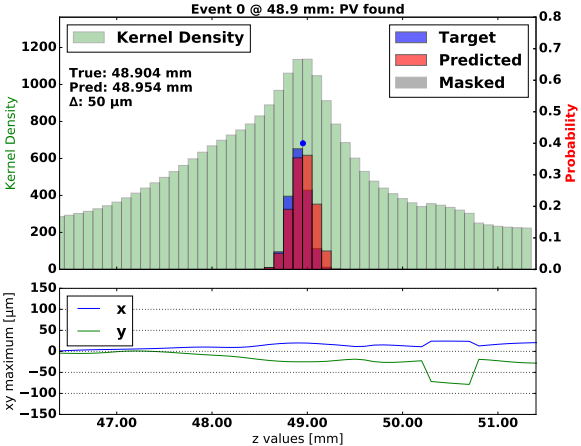
Search for PVs

- Search ± 5 bins ($\pm 500 \mu$m) around a true PV
- At least 3 bins with predicted probability $> 1\%$ and integrated probability $> 20\%$.

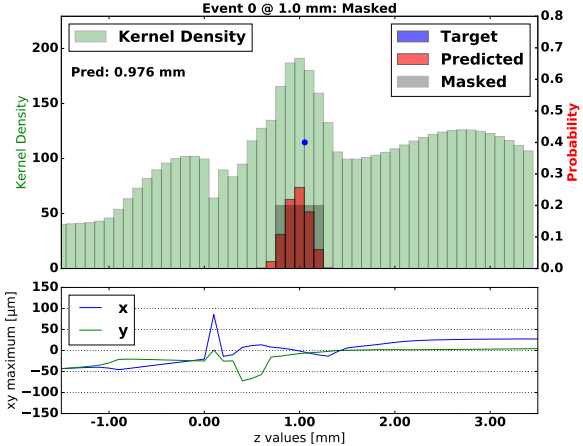
Tunable efficiency vs. FP

- The asymmetry parameter controls FP vs. efficiency

Compare predictions with targets: When it works

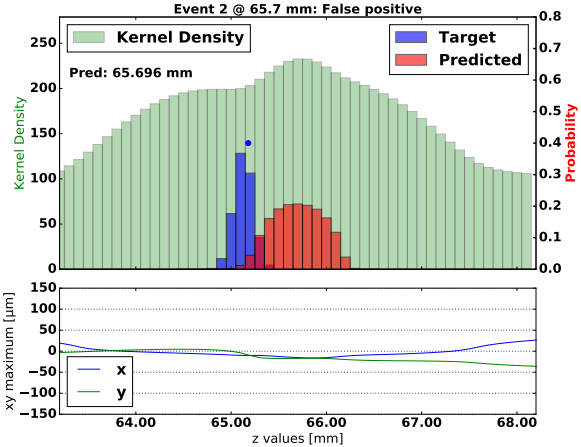


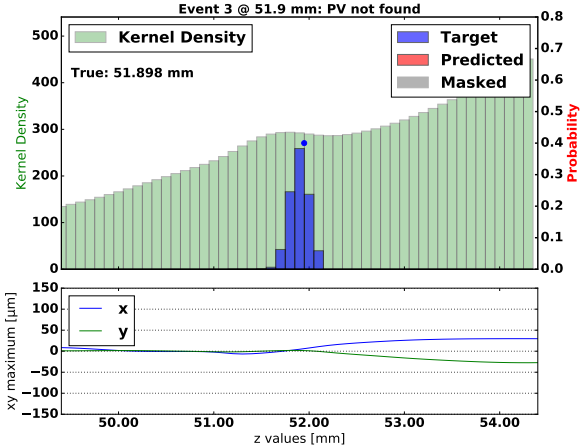
PV found example



Masked (<5 tracks) example

Compare predictions with targets: When it fails



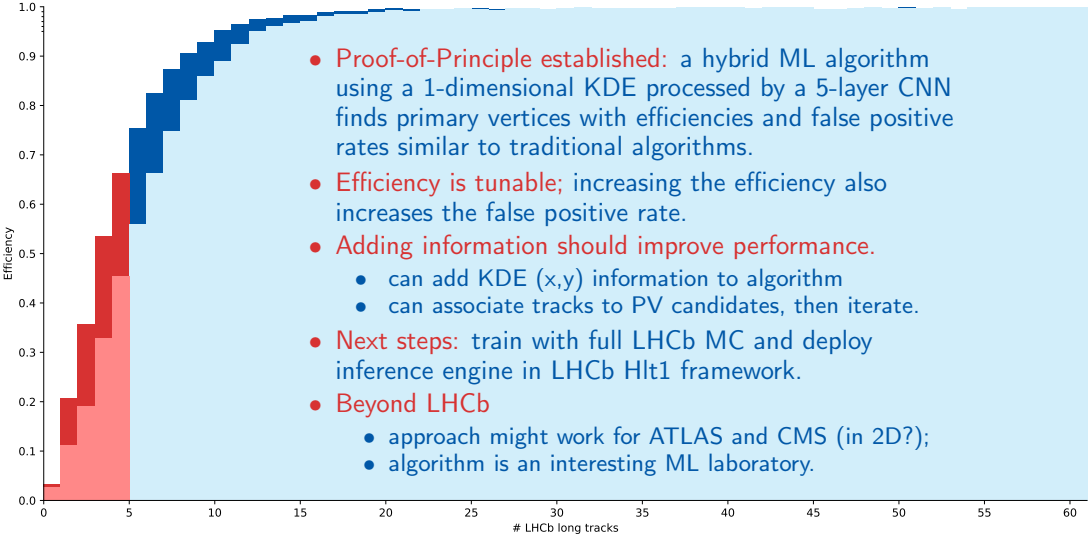


False Positive example

PV not found example

Conclusions and plans

Future plans



- Proof-of-Principle established: a hybrid ML algorithm using a 1-dimensional KDE processed by a 5-layer CNN finds primary vertices with efficiencies and false positive rates similar to traditional algorithms.
- Efficiency is tunable; increasing the efficiency also increases the false positive rate.
- Adding information should improve performance.
  - can add KDE (x,y) information to algorithm
  - can associate tracks to PV candidates, then iterate.
- Next steps: train with full LHCb MC and deploy inference engine in LHCb Hlt1 framework.
- Beyond LHCb
  - approach might work for ATLAS and CMS (in 2D?);
  - algorithm is an interesting ML laboratory.

Questions?

Source code:

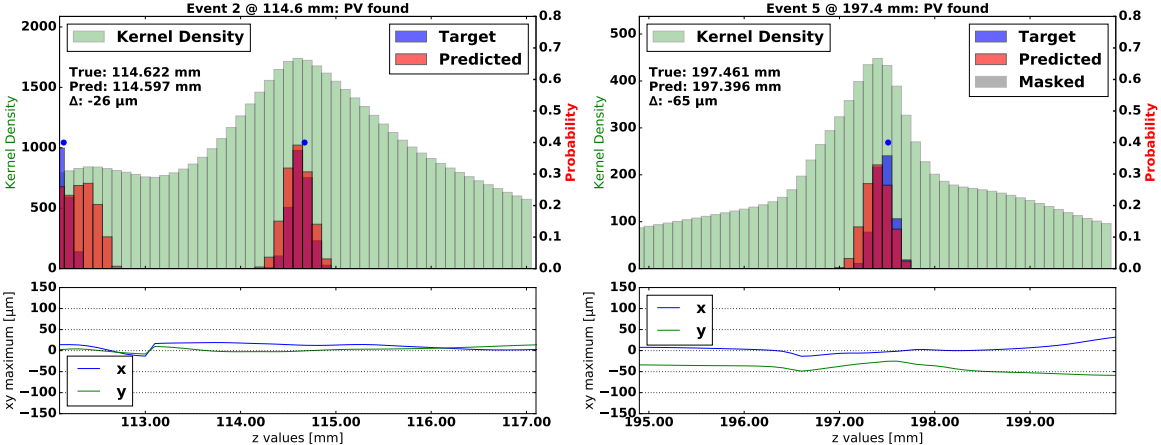
- https://gitlab.cern.ch/LHCb-Reco-Dev/pv-finder
- Runnable with Conda on macOS and Linux
  - Run: `conda env create -f environment-gpu.yml`
  - Python 3.6+ and PyTorch used for machine learning code
  - Generation now available too using the new Conda-Forge ROOT and Pythia8 packages

Supported by:

- NSF OAC-1836650: IRIS-HEP
- NSF OAC-1740102: SI2:SSE
- NSF OAC-1739772: SI2:SSE

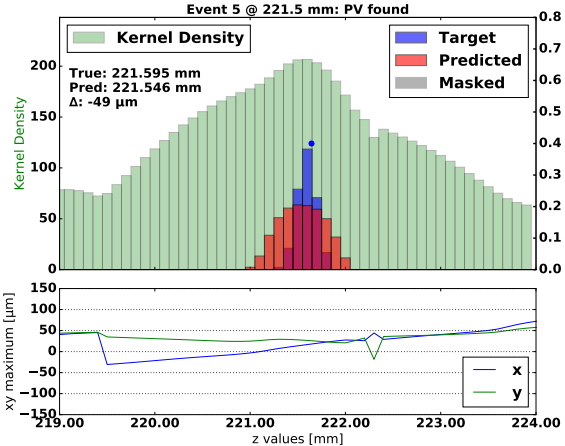
More predictions with targets (1)

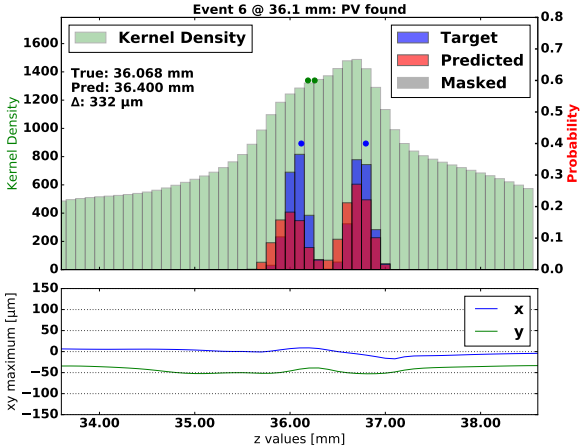
Backup



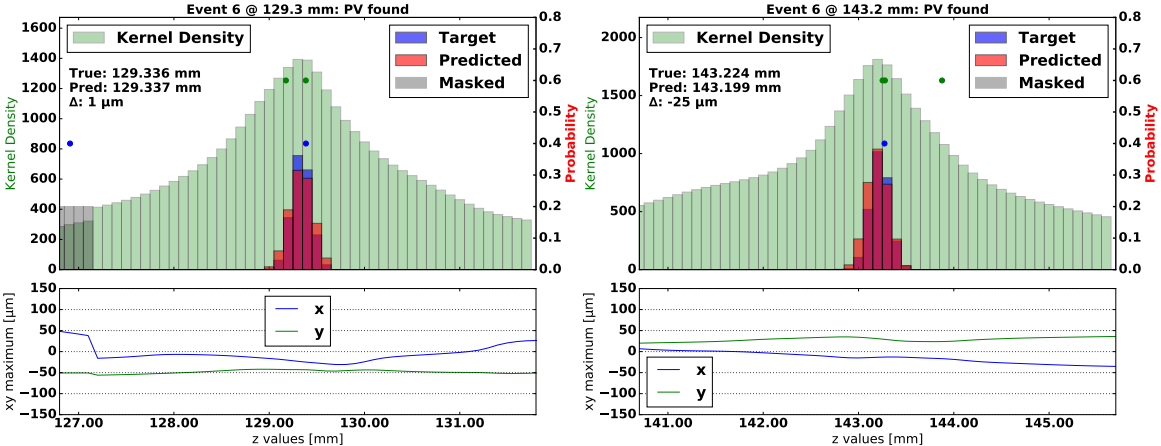
More predictions with targets (2)

Backup

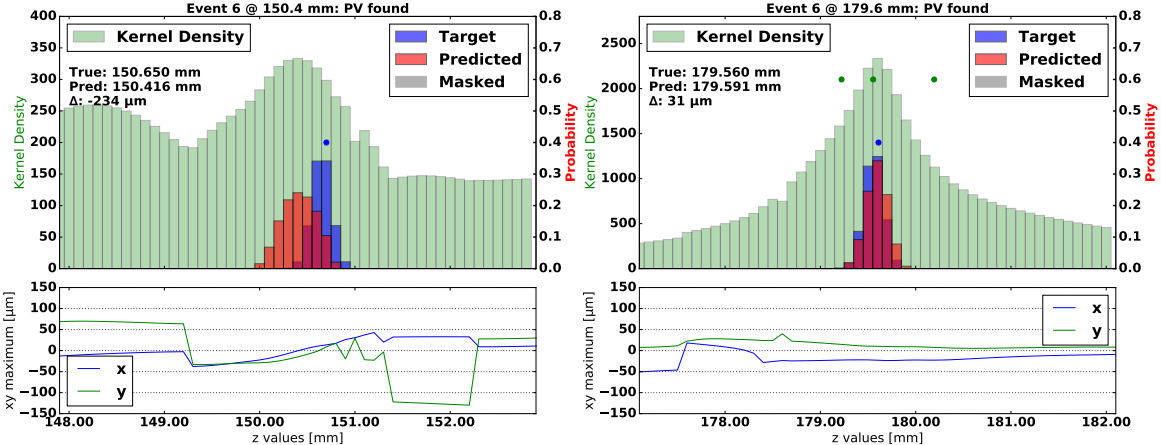




More predictions with targets (3)



More predictions with targets (4)

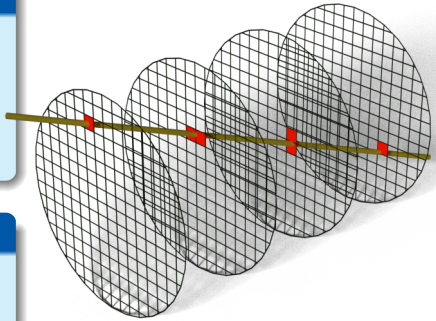


Tracks

- Originate from vertices (not shown)
- Hits originate from tracks
- We only know the true track in simulation
- Nearly straight, but tracks may scatter in material

The VELO

- A set of 26 planes that detect tracks
- Tracks should hit one or more pixels per plane
- Sparse 3D dataset (41M pixels)



Questions for other experiments

- Beam width (x, y): $40\,\mu\mathrm{m}$ for LHCb, what is yours?
- Transverse resolution: 5–15 $\mu\mathrm{m}$ for LHCb depending on number of tracks, what is yours?
- Longitudinal resolution: 40–100 $\mu\mathrm{m}$ for LHCb depending on number of tracks, what is yours?
- Cleaning up prototracks based on IP could simplify kernel
- Can prototracking be done in the triggers?