

# Charged Particle Track Finding with MCTS

Max Zhao (UC Berkeley)

Mentors:

Johannes Wagner

Louis-Guillaume Gagnon

Heather Gray

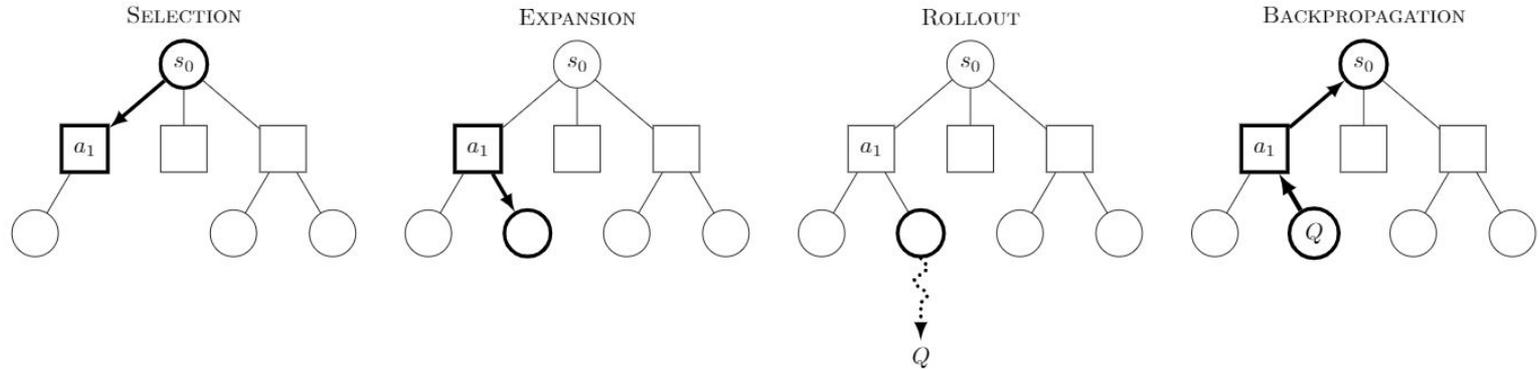


# Motivation

- Original goal of the project was to investigate track reconstruction algorithms that integrate machine learning and Kalman filtering.
  - Kalman filtering for domain knowledge.
  - Machine learning for improved GPU architecture.
- Something similar in structure to the Combinatorial Kalman Filter (CKF) but with strong machine learning heuristics.

# Monte Carlo Tree Search (MCTS)

Algorithm that uses random playouts for optimal action selection. Edges of a tree are iteratively updated with these playouts to bring out “promising” actions.



# Structural Advantages of MCTS for tracking

- Kalman filtering can be easily integrated into prior probability calculations.
  - Provides additional information like momentum which can be fed into the Policy network.
- MCTS aggressively prunes the search space, which could improve scalability of algorithm.
- MCTS iteratively reduces uncertainty, so it can be adapted to different requirements depending on the stage of reconstruction.

# Neural Networks

## Policy network

- Takes in a pair of hits on adjacent layers and returns how likely they are to be part of the same track.

## Evaluation network

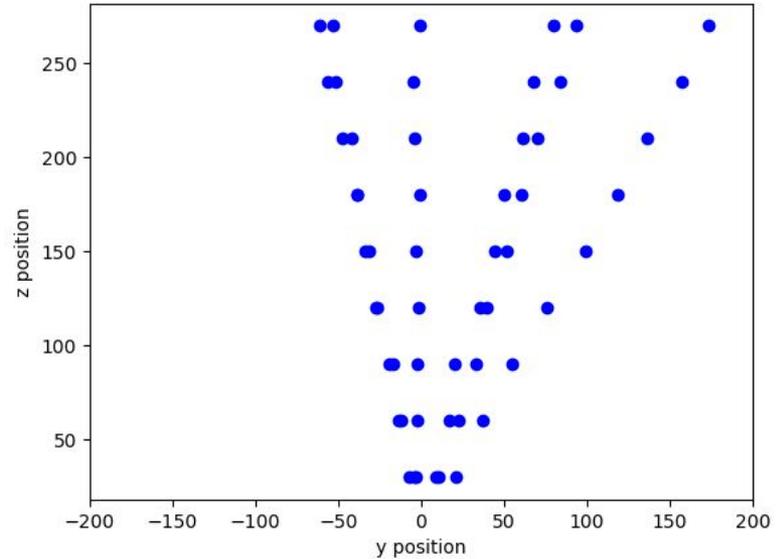
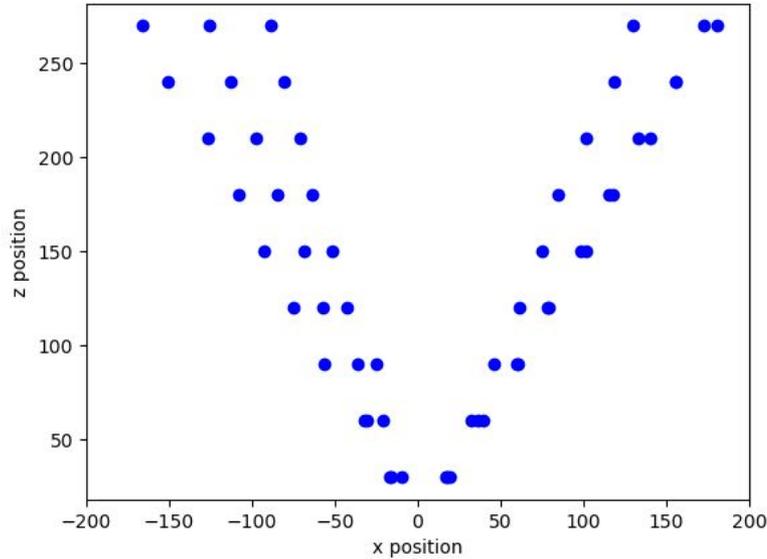
- Takes in a list of hits in sequence and returns how likely it is to be a valid track.

# Algorithm Description

For each iteration:

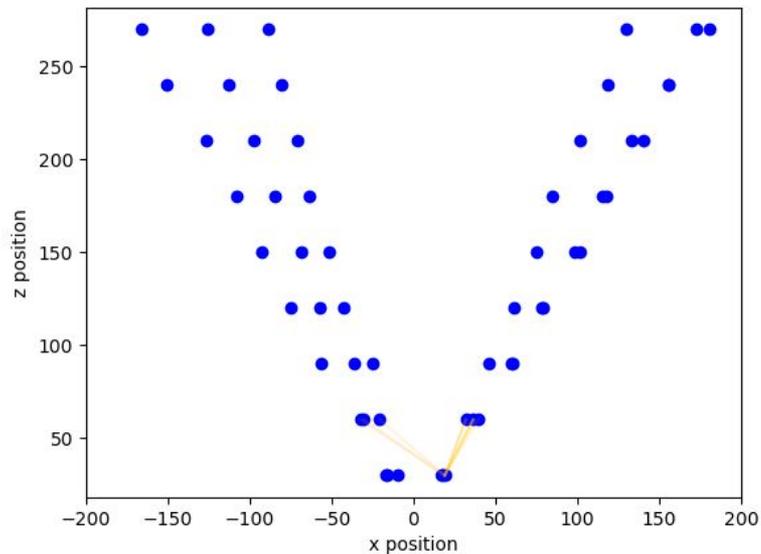
1. Traverse the tree to find an unexplored “promising” hit.
  - a. Traversal is done with prior probabilities, previous playouts, and possibly Kalman filtering
2. Expand the hit to new options, imbuing each with a prior probability from the policy network.
3. Execute a Monte Carlo playout with the policy network from that hit constructing a track candidate.
4. Evaluate the candidate with the evaluation network and update all visited edges.

# Example (9 layer telescope detector, 8 muon event)

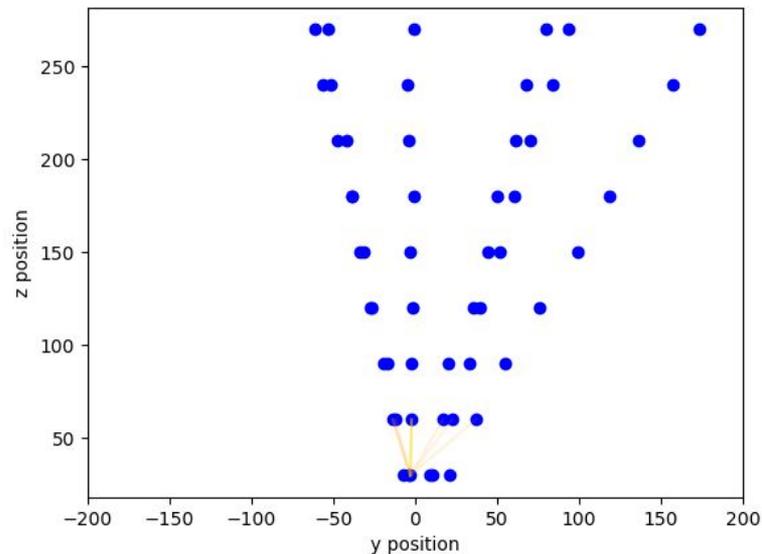


# Iteration 1

Iteration 1

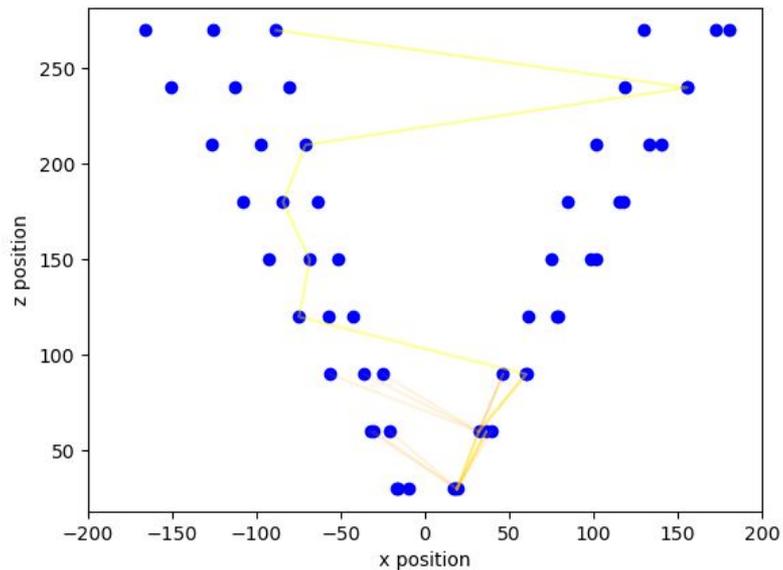


Iteration 1

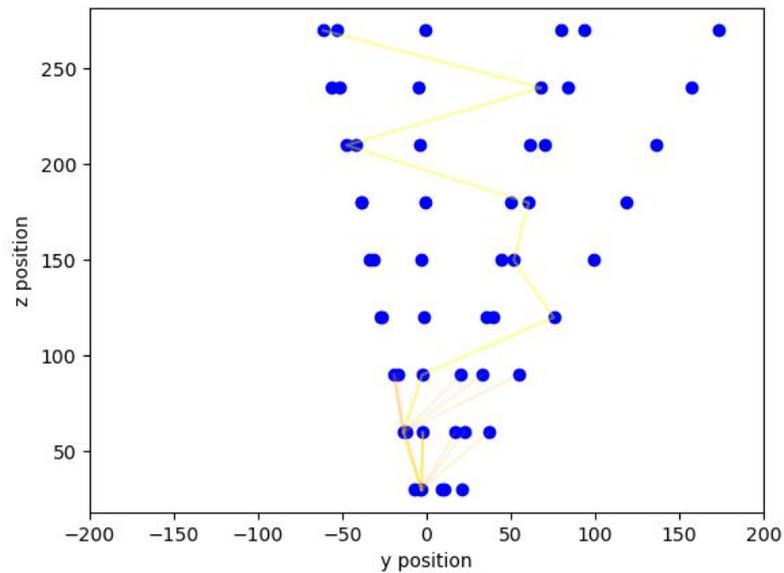


# Iteration 2

Iteration 2

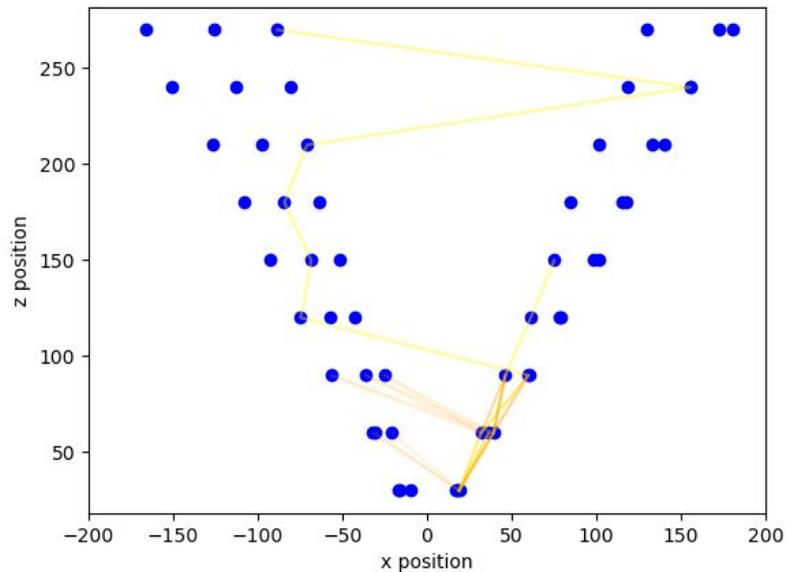


Iteration 2

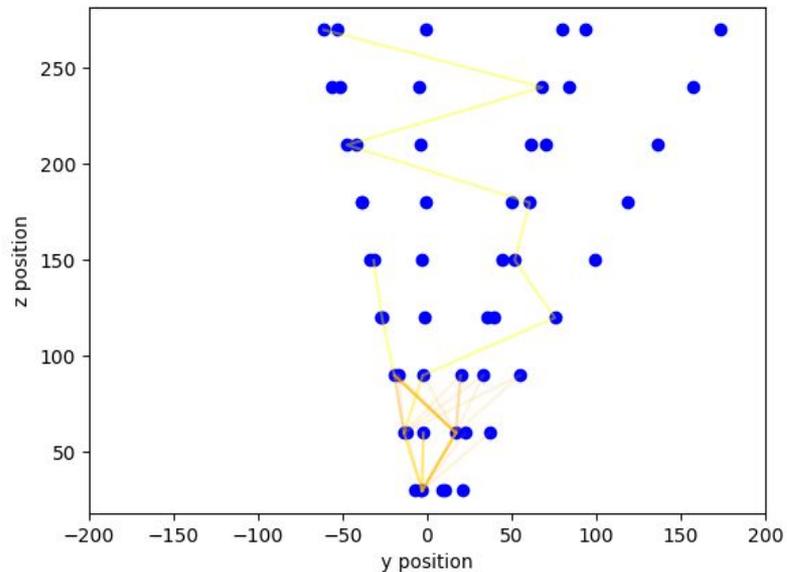


# Iteration 3

Iteration 3

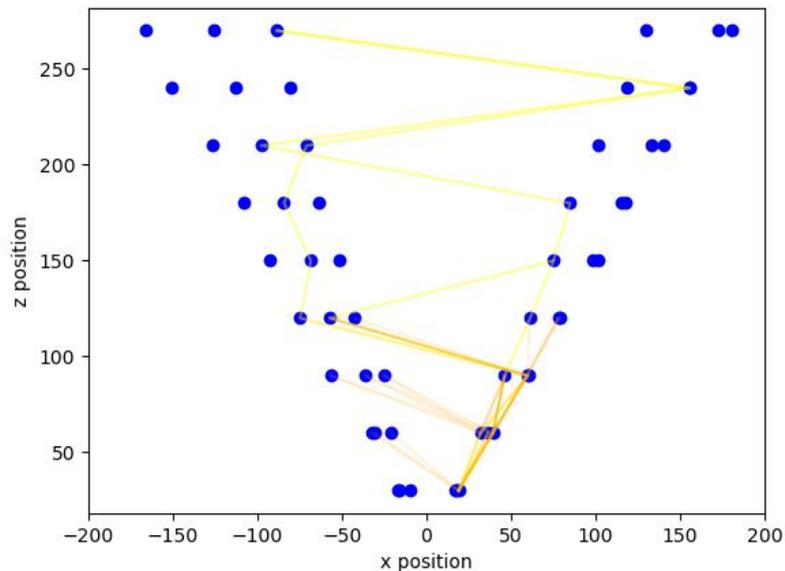


Iteration 3

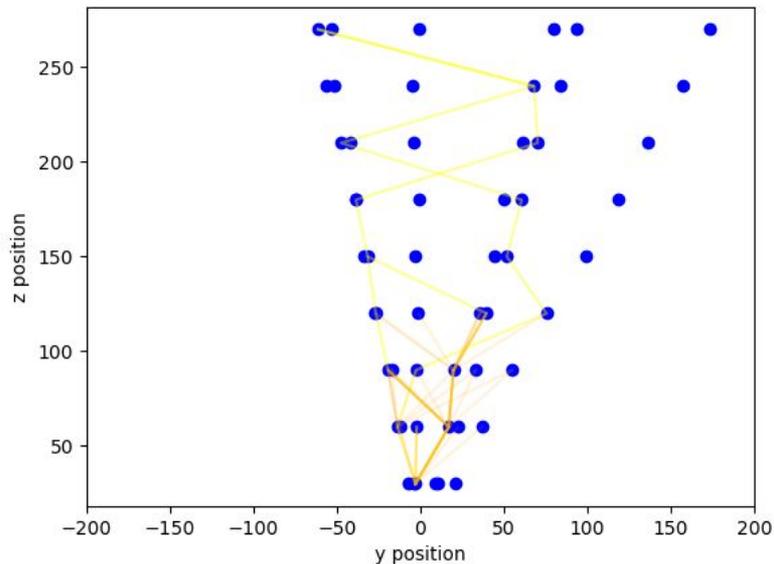


# Iteration 4

Iteration 4

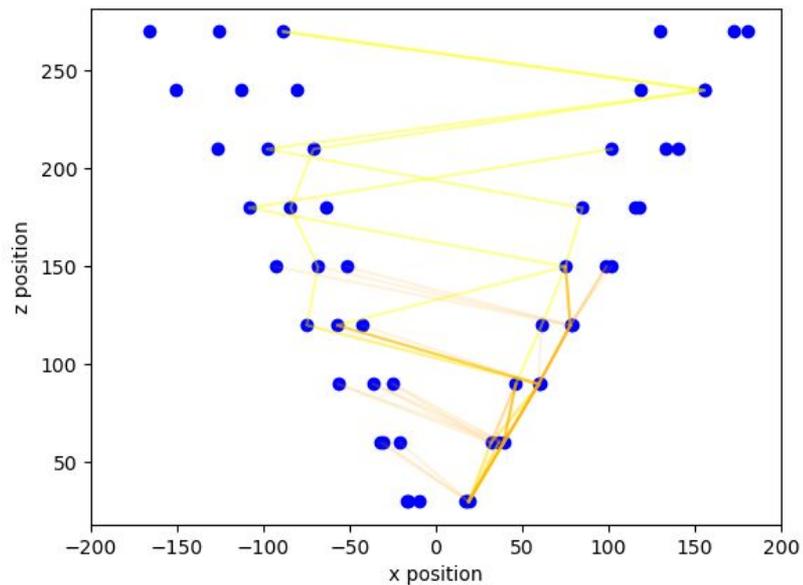


Iteration 4

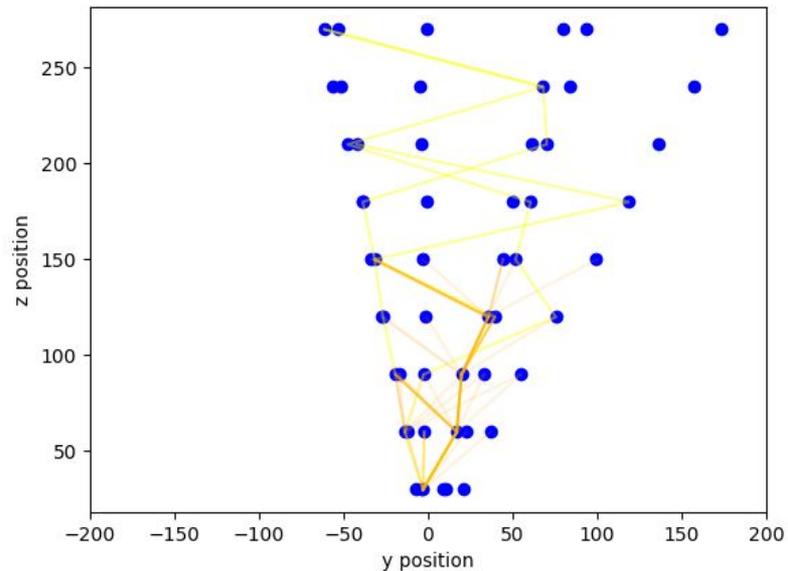


# Iteration 5

Iteration 5

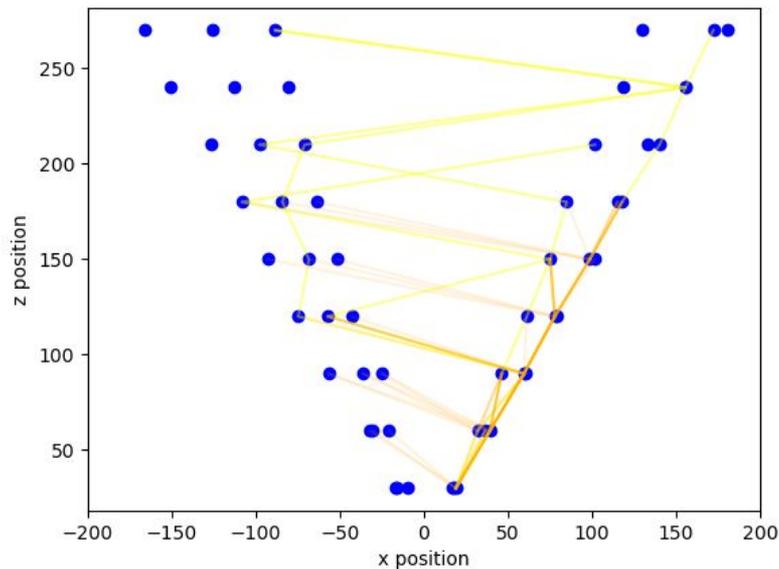


Iteration 5

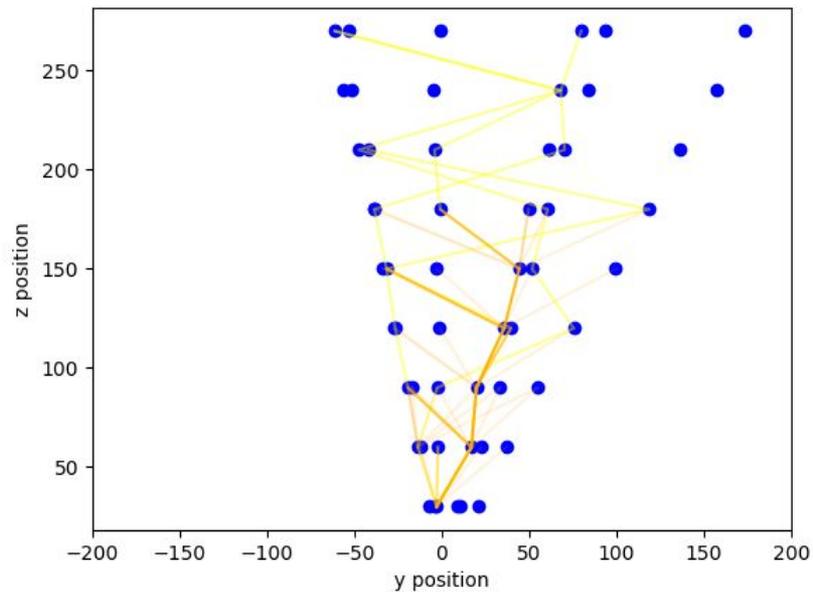


# Iteration 6

Iteration 6

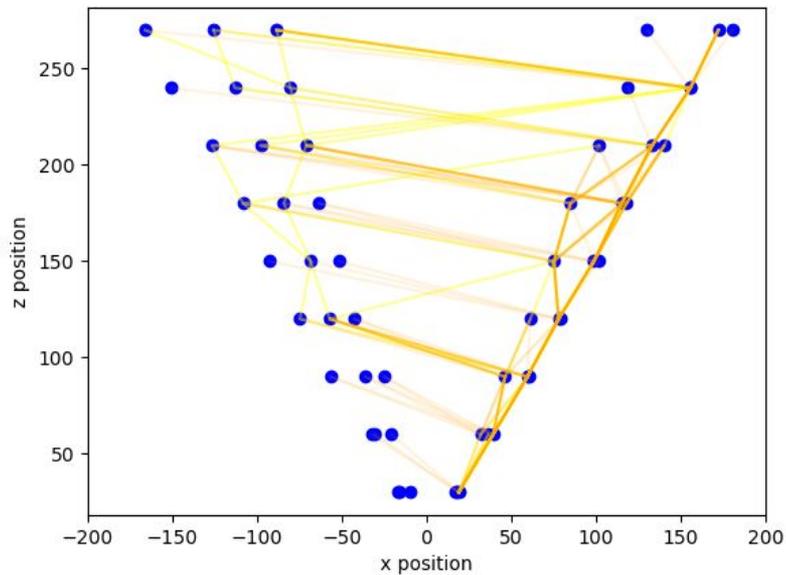


Iteration 6

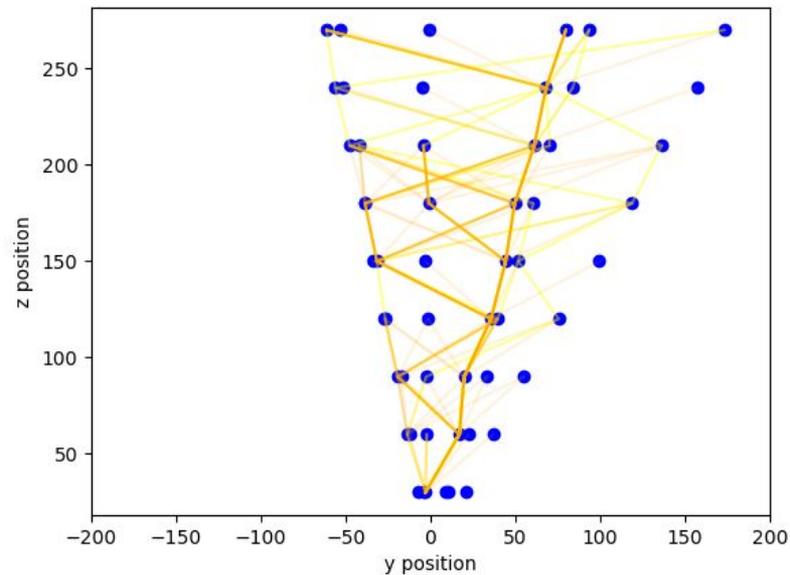


# Iteration 16

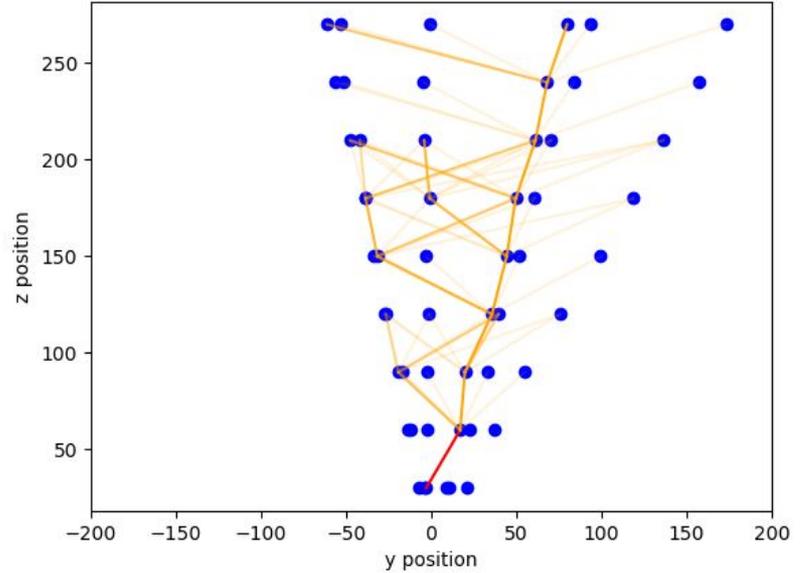
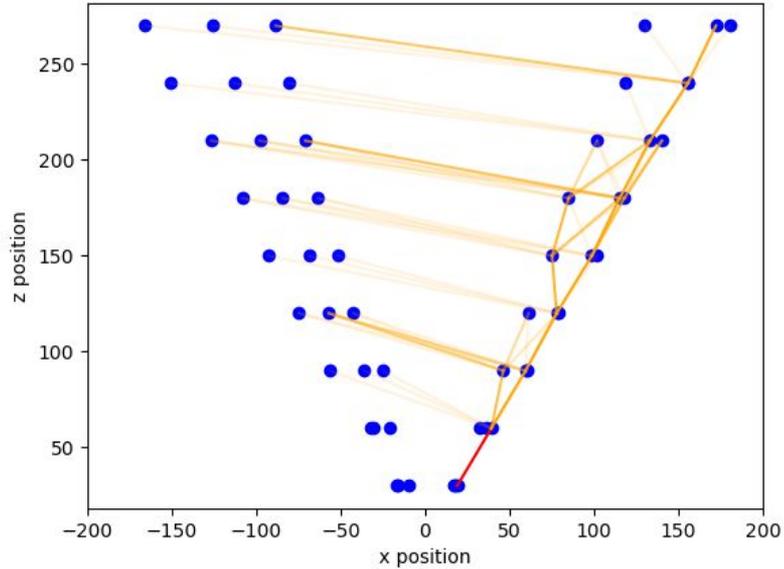
Iteration 16



Iteration 16



# Locking in an edge



# Preliminary Testing

- MCTS is seen to augment performance when policy is weak.
- No effect or even degradation when policy is strong.

	Random policy	Trained policy
Policy only efficiency	42.0%	99.6%
Policy only purity	69.3%	99.6%
MCTS efficiency	66.8%	98.9%
MCTS purity	77.6%	99.2%

# Improvements and Future directions

- Algorithm currently does not consider tracks skipping layers. This will need to be added as another possible action.
- Policy network can be split into two networks, one fast for MC playout and one slow for assigning prior probabilities.
- Kalman filtering should be integrated into tree traversal.
- Goal is to apply algorithm to more sophisticated detector geometry and higher density events.
- Develop more precise benchmarks for comparison with other algorithms.

# Questions