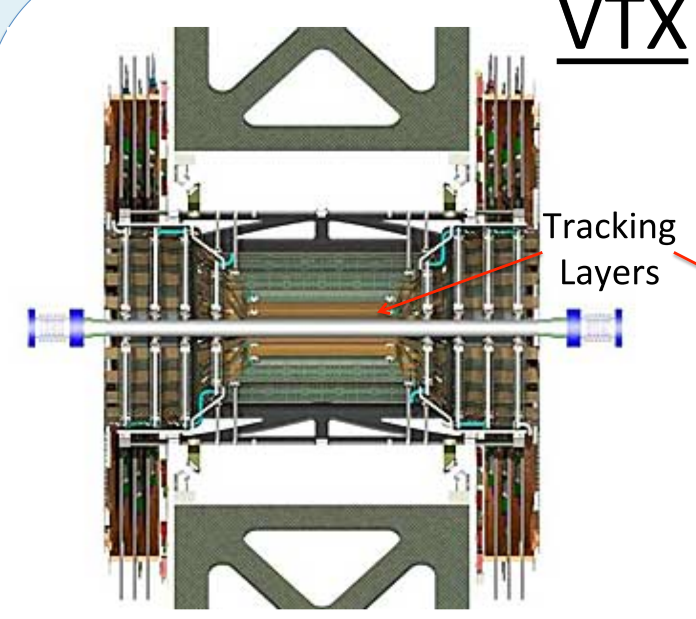
Two-Particle Correlation Results from the PHENIX Silicon Vertex Detector in Au+Au collisions at 200 GeV

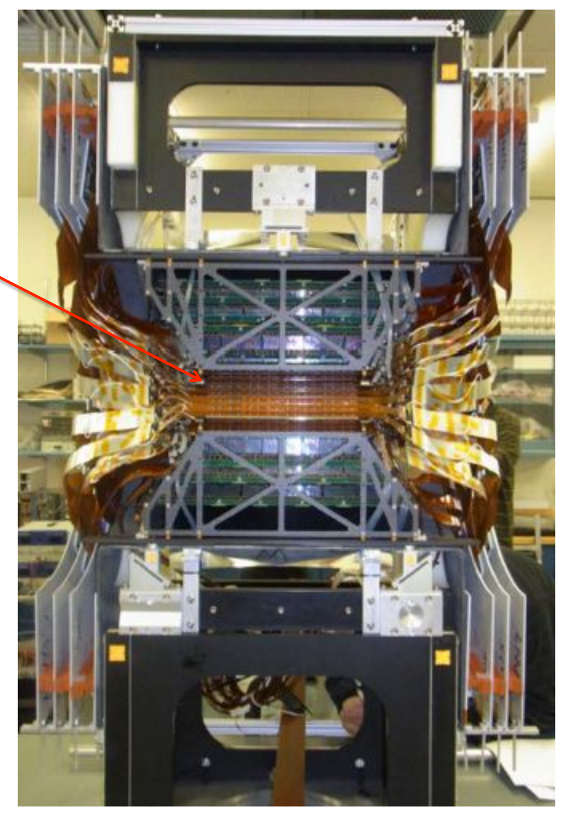
by Theo Koblesky (University of Colorado) for the PHENIX Collaboration

Motivation

The VTX (Silicon Vertex Detector) is a recent detector upgrade to PHENIX which extends the rapidity coverage out to ±1 unit. This coverage is useful for a two-particle correlation analysis using charge particle tracks through the VTX only. sPHENIX is a majoy upgrade that includes increasing the tracking layers from 4 to 6. For both cases, a robust tracking algorithm is required in order to make these measurements with either the VTX or sPHENIX. A novel and promising tracking algorithm which makes use of the Hough Transform developed by Alan Dion is available publically at Ref [1].

VTX

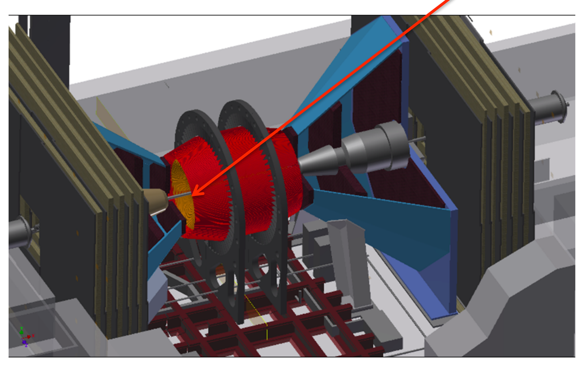


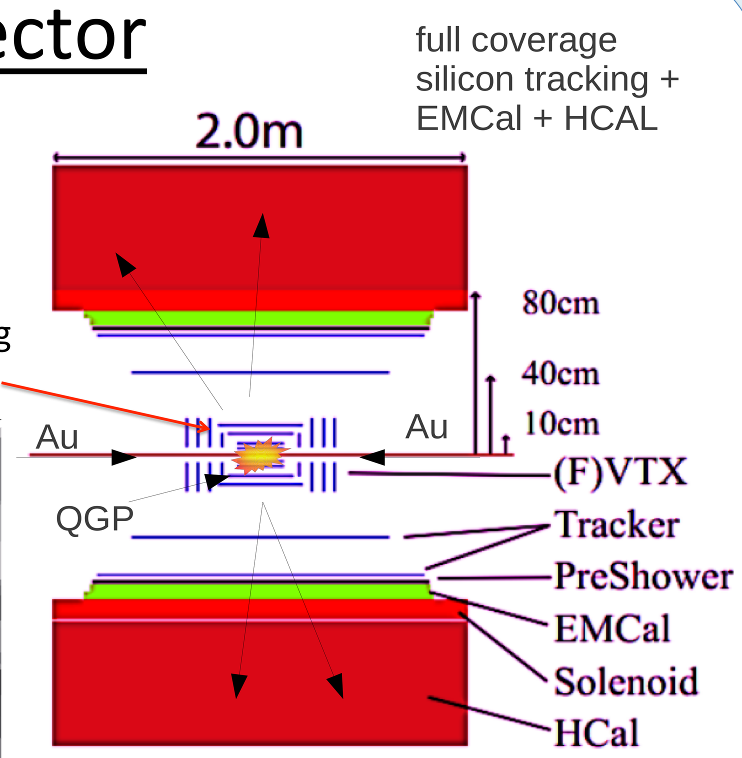


The VTX is composed of 2 inner layers of silicon pixels and 2 outer layers of silicon strip-pixels.

sPHENIX Detector

The proposed sPHENIX upgrade has 6 layers for tracking over the full azimuth.

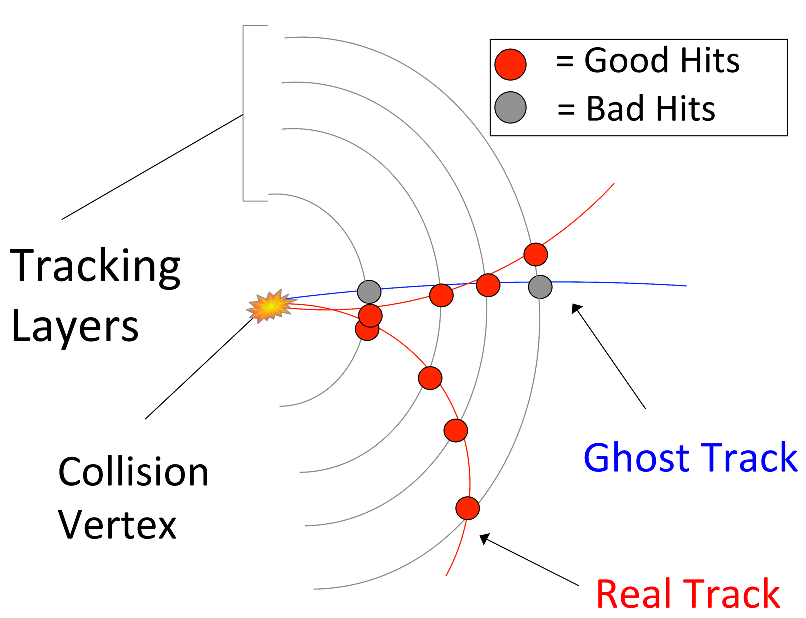




Introduction to Tracking

The basic goal of tracking is to associate detector hits to reconstruct true particle tracks. Ghost tracks are false associations made by the tracking algorithm which result in a reconstructed track not based on a true particle track. These tracks can be reduced through chi squared minimization filtering or similar methods. The goal of any tracking algorithm is to minimize the number of ghost tracks and maximize the number of real tracks found, while making maximal efficiency of computational resources such as CPU & memory.

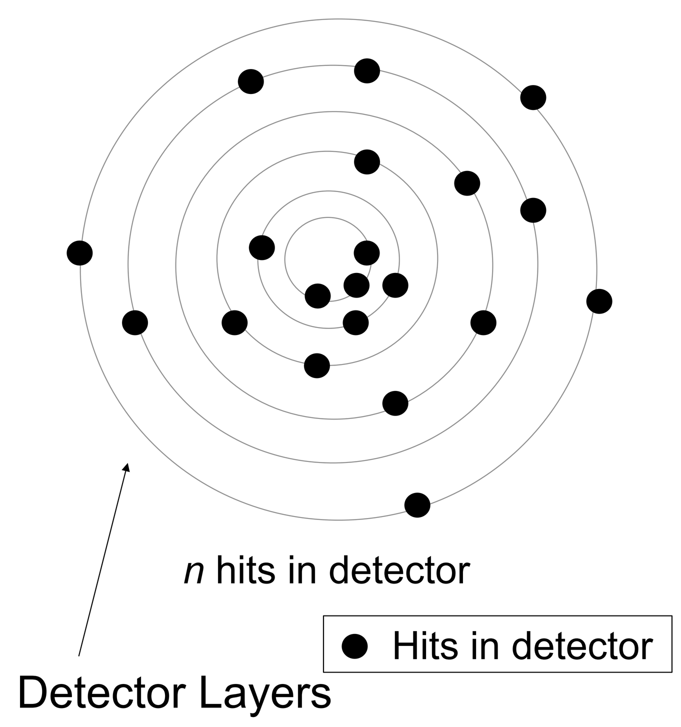
Tracking example in a 4 layer detector with constant magnetic field



Brute Force Tracking

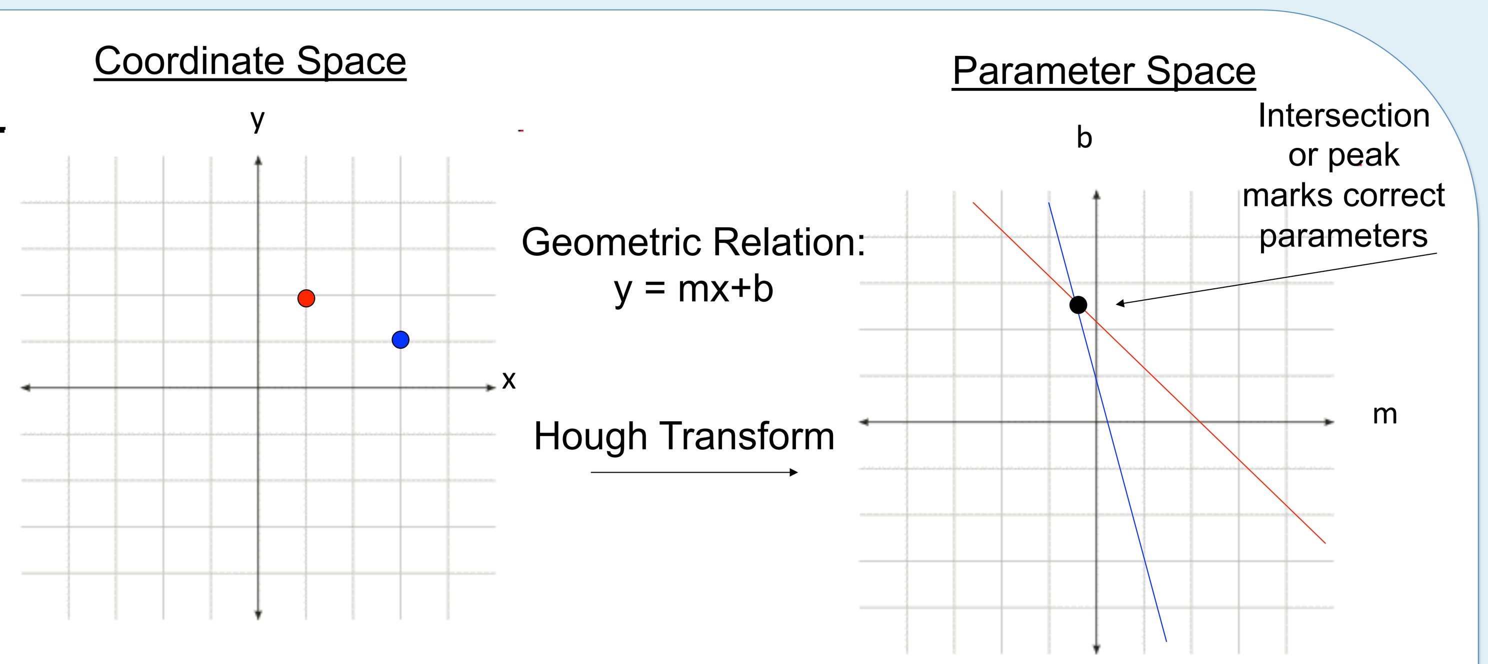
The simplest and most efficient tracking algorithm possible is to make track candidates out of every possible combination of hits in the detector. This method guarantees to reconstruct every real track passing through the detector, as well as a large number of fake tracks. While brute force tracking may be crude, it is useful as a baseline which other tracking algorithms can be compared with. For example, the asymptotic scaling of brute force tracking is $O(n^{\text{layers}})$ in big O notation. More sophisticated tracking algorithms will want to reduce the asymptotic scaling while maintaining high track finding efficiency.

Example of tracking in a full azimuth 6 layer detector (like sPHENIX)



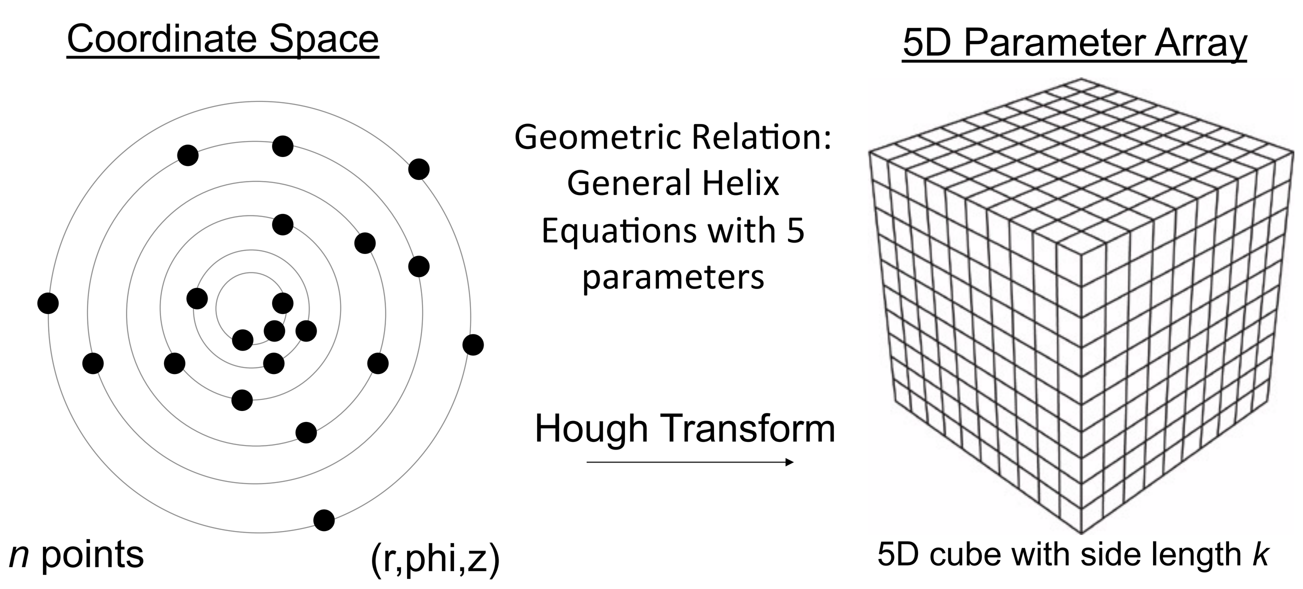
The Hough Transform

The Hough Transform takes a set of points and an unknown geometric relation between them and transforms them from Coordinate Space to Parameter Space. For helical charged tracks, cylindrical coordinates is the Coordinate Space and the geometric relation between the points is a helix.



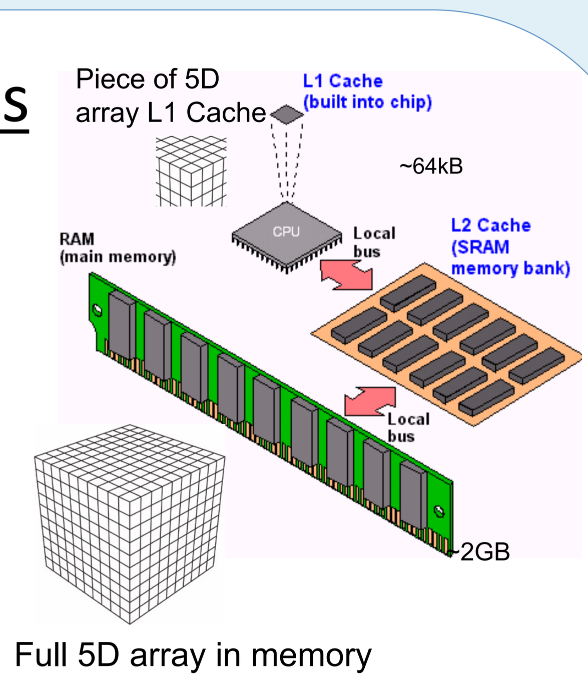
Hough & Tracking

The Hough Transform can be used for tracking by finding peaks in the Parameter Space. Bins with peaks indicate that there is a possible track candidate in that bin. Then another tracking algorithm is applied to the bin until every possible good track is extracted. In order to find peaks in the Parameter Array, we need the binning to be at least $k \approx \sqrt{n}$.

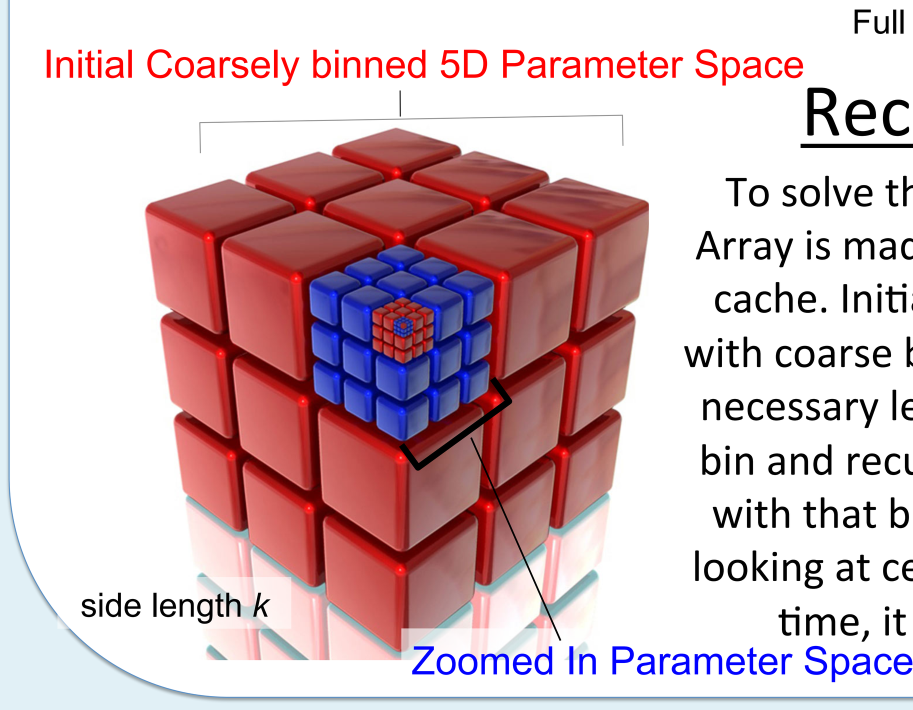


Computational Limitations

The required fineness of binning of the 5D Parameter Array makes it too large to fit into the CPU's local cache. Random accesses of the array forces continual filling of the cache with pieces of the array stored in memory. Not fitting the entire array into the CPU cache slows down the algorithm considerably.



Recursive Zooming



To solve this problem, the Parameter Array is made so that it always fits in the cache. Initially the array is constructed with coarse binning. In order to get to the necessary level of $k \approx \sqrt{n}$, we find a peak bin and recursively create another array with that bin or "zoom into it". By only looking at certain pieces of the array at a time, it always fits in the cache.

Algorithmic Performance

The asymptotic scaling can be calculated and compared with the Brute Force algorithm. To calculate the asymptotic scaling for the Hough algorithm, we assume k^3 iterations to Hough Transform a single point. For n points at sufficient binning, this operation takes $n^{5/2}$ iterations. To account for the recursive zooming, add a factor of log(n). The Hough has better scaling than the brute force.

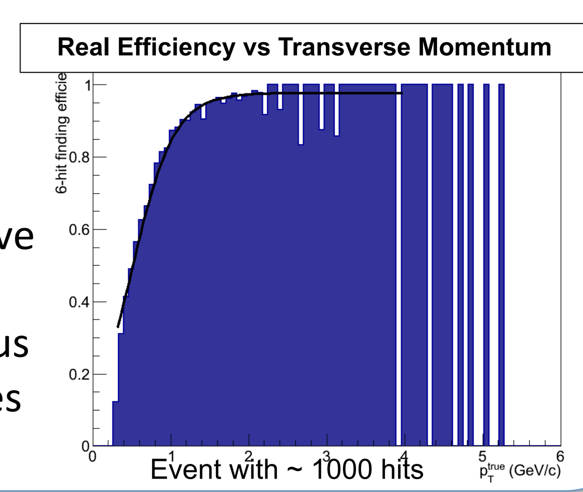
Asymptotic Scaling Table

n ~ hits L = layers	Brute Force	Cellular automata*	Hough Algorithm
No vertex info	**$O(n^{L})$**	**$O(n^2)$**	**$O(n^{5/2}\log(n))$**
With Vertex info	**$O(n^{L-1})$**	**$O(n^2)$**	**$O(n^{3/2}\log(n))$**

*cellular automata implemented by CBM and STAR past year

Track Finding Efficiency

The efficiency of the algorithm is calculated by taking the number of correctly reconstructed tracks over the number of tracks which could have been reconstructed. For a simulation throwing 1000 tracks per event in a 6 layer detector, versus transverse momentum the efficiency asymptotes to 97% by 2 GeV/c.



Thanks to Alan Dion for the development of the algorithm. Thanks to Mike McCumber for his invaluable help in deciphering and studying this algorithm. Thanks to the Department of Energy for their generous grant support.

Ref[1]: code.google.com/p/helixhough/