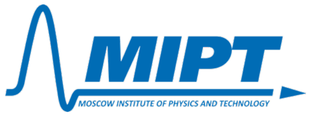
Yandex



Track Finding

Mikhail Hushchyn, Andrey Ustyuzhanin

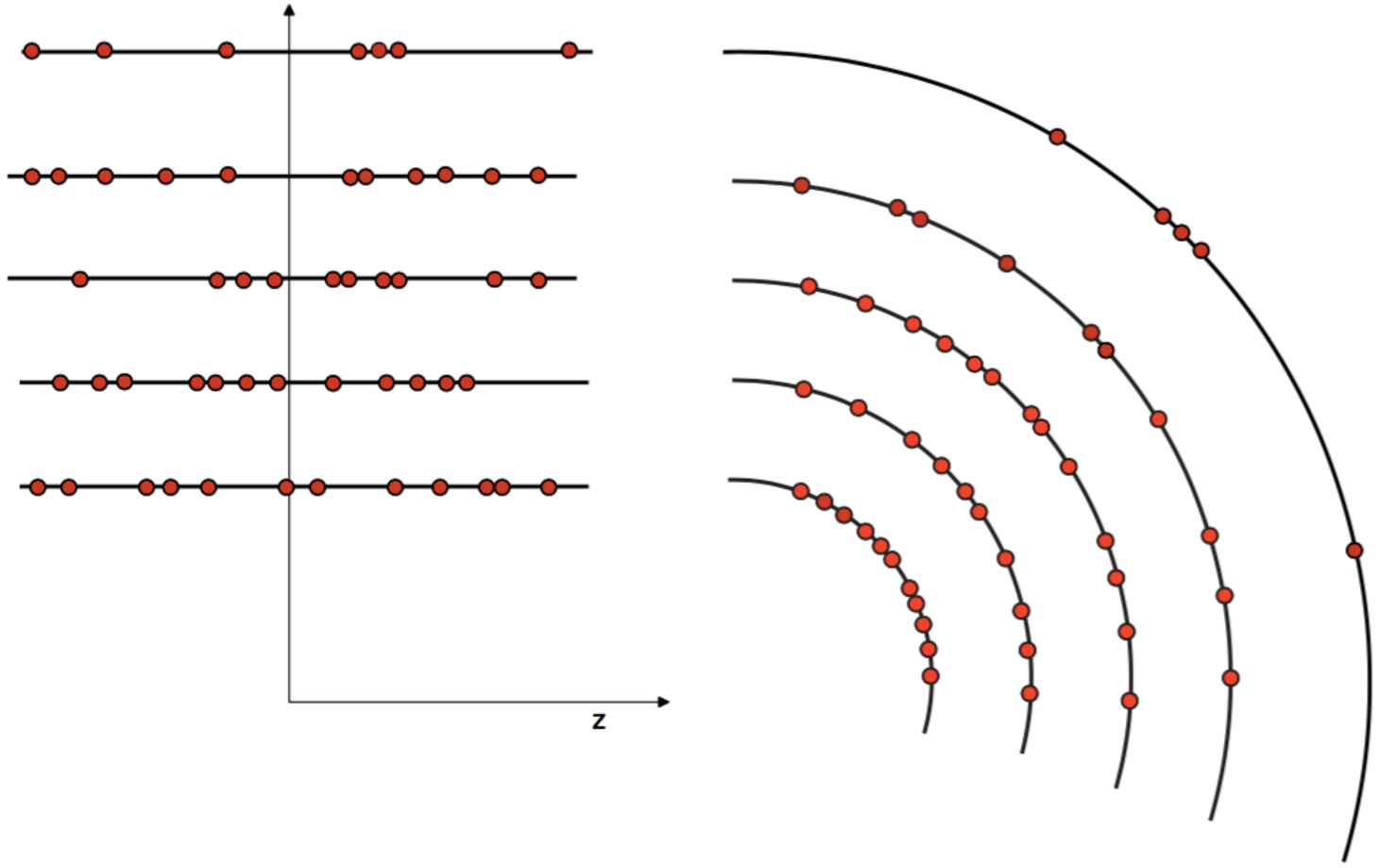
Outline

1. Problem formulation
2. Global methods
3. Local methods

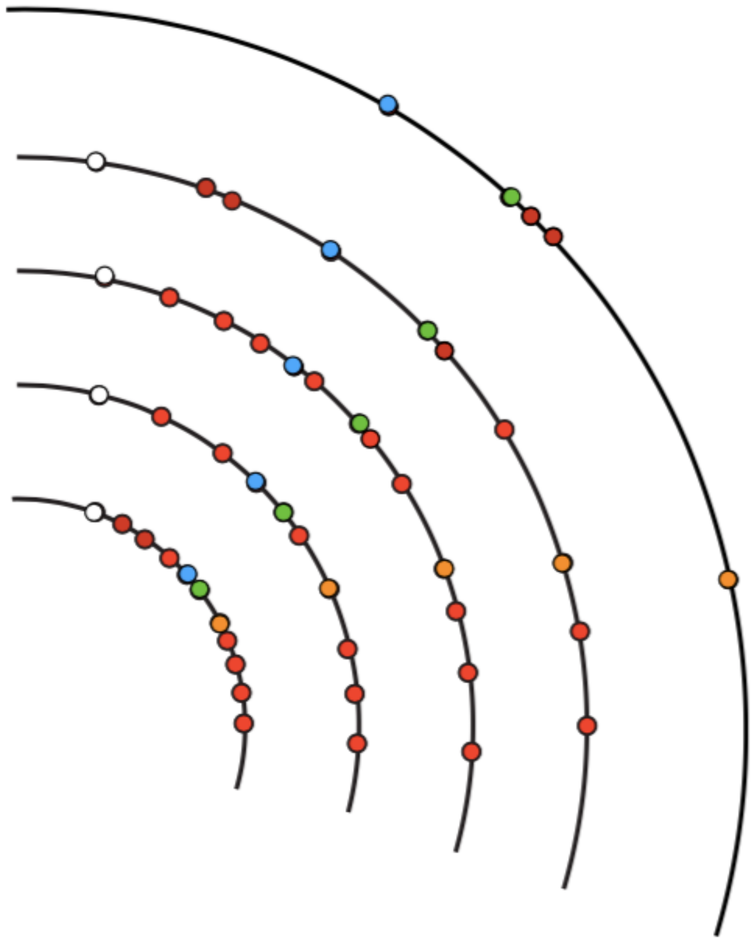
Track Finding

Problem formulation

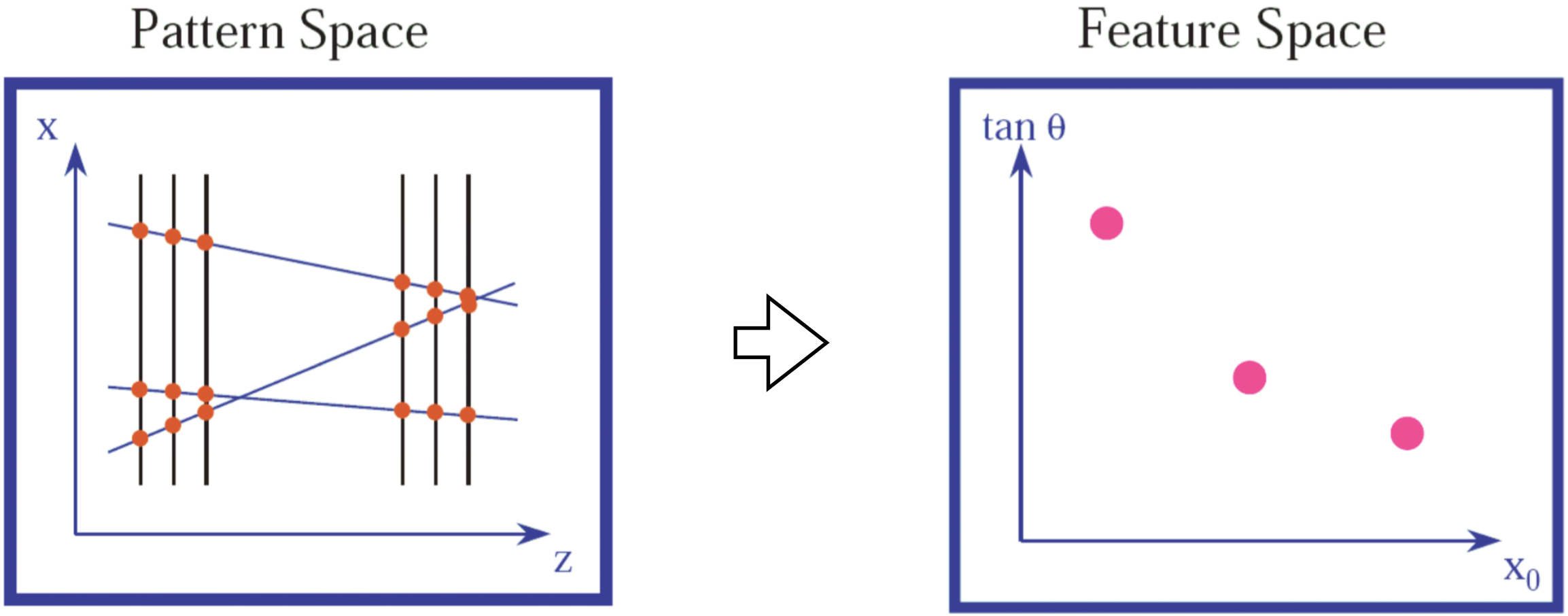
Finding tracks



Finding tracks



Tracks finding and fitting



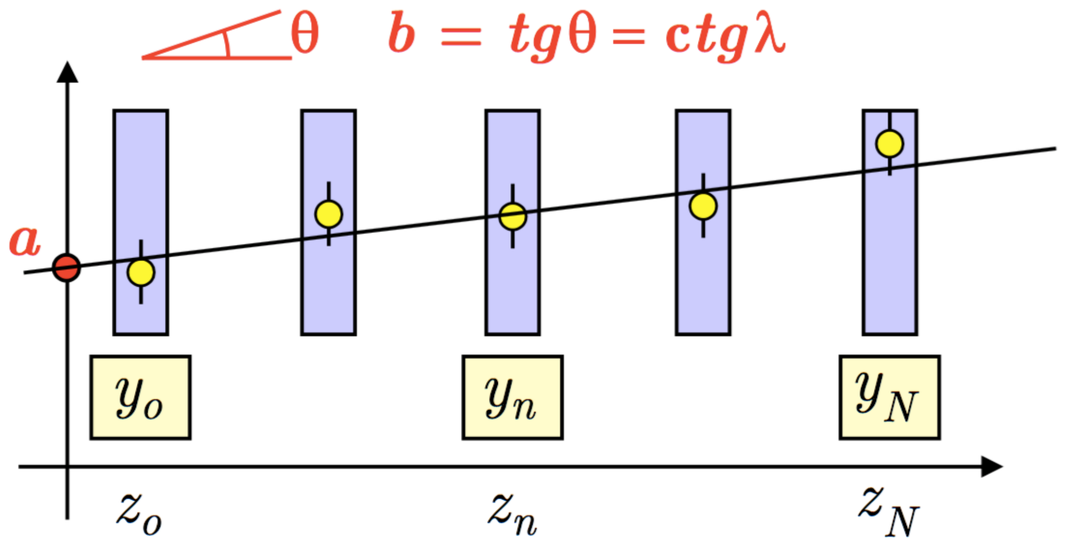
Track Finding

Global methods

Template matching

Template matching

$y = a + b\,z$



$$\chi^2 = \sum_{n=0}^{N} \frac{(y_n - a - bz_n)^2}{\sigma_n^2}$$

Template matching

The χ^2 can be written as

$$\chi^2 = (\mathbf{Y} - \mathbf{Ap})^T \mathbf{W} (\mathbf{Y} - \mathbf{Ap}) \qquad \mathbf{W} = \mathbf{V}^{-1}$$

The minimum χ^2 is obtained by

$$\tilde{\mathbf{p}} = \left(\mathbf{A}^T \mathbf{W} \mathbf{A}\right)^{-1} \mathbf{A}^T \mathbf{W} \mathbf{Y}$$

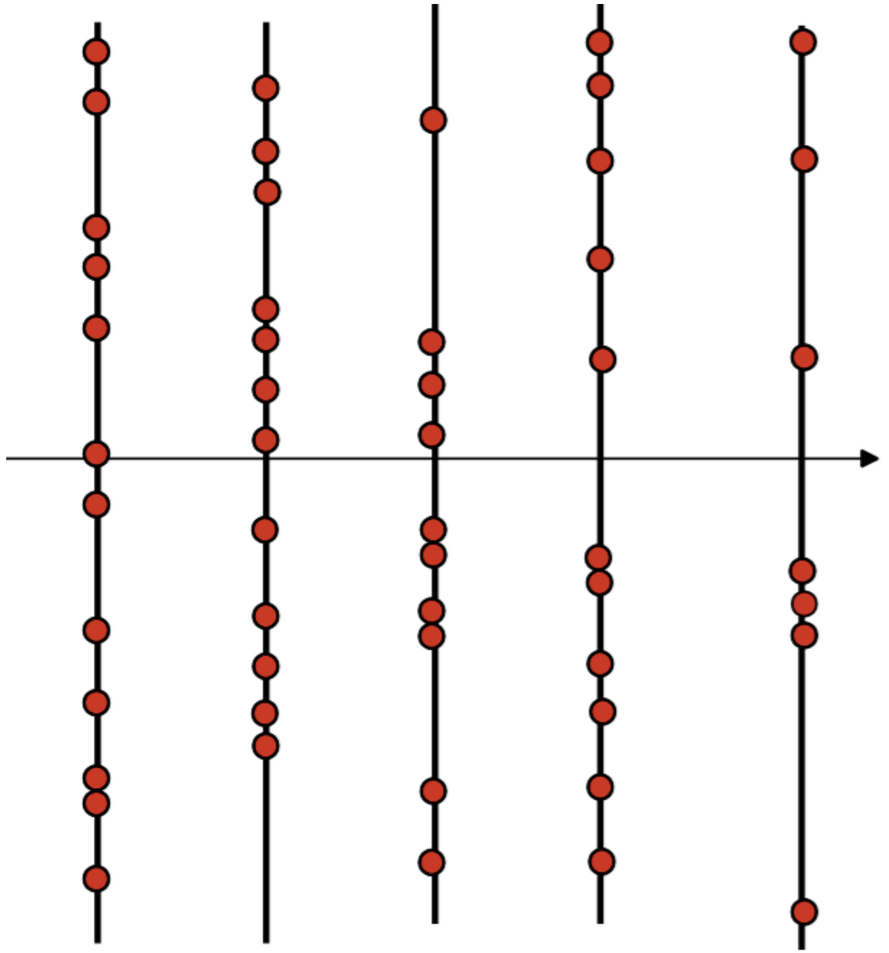
The covariance matrix of the parameters is obtained from the measurements covariance matrix V

$$\mathbf{V_P} = \left(\mathbf{A}^T \mathbf{V}^{-1} \mathbf{A}\right)^{-1}$$

Template matching

1. Make complete list of 'patterns', valid combinations of hits
2. Now simply run through list of patterns and check for each if it exists in data

This works well if number of patterns is small.

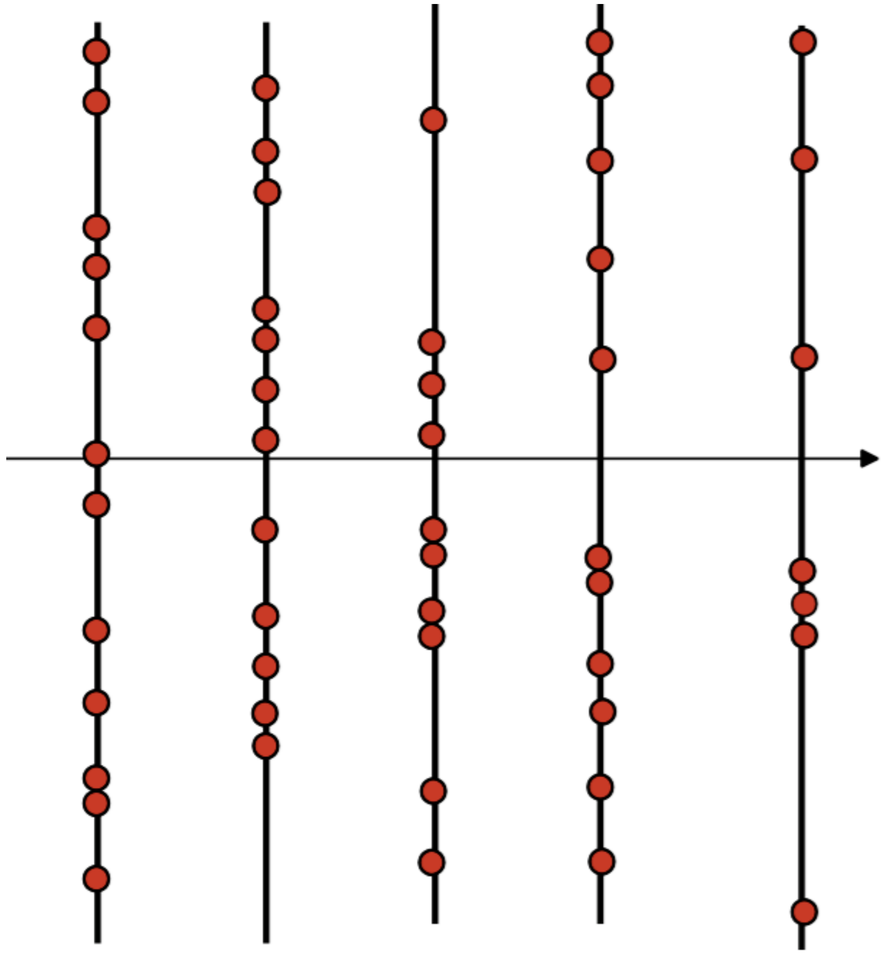


Template matching

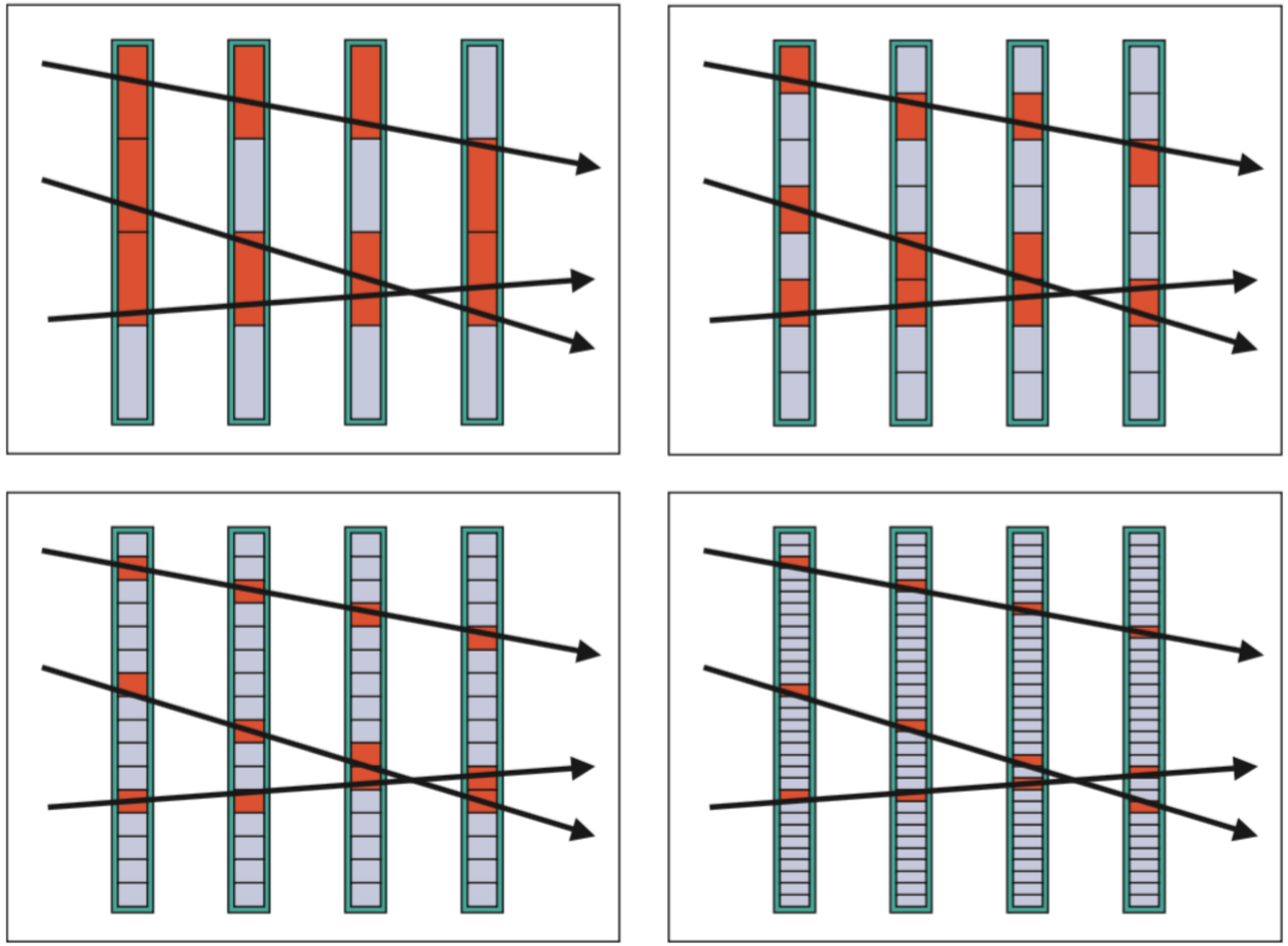
Difficulties:

〉 Does not scale very well when the problem requires high dimensionality or granularity

〉 The number of computations increases significantly with a finer resolution of templates.



Template matching



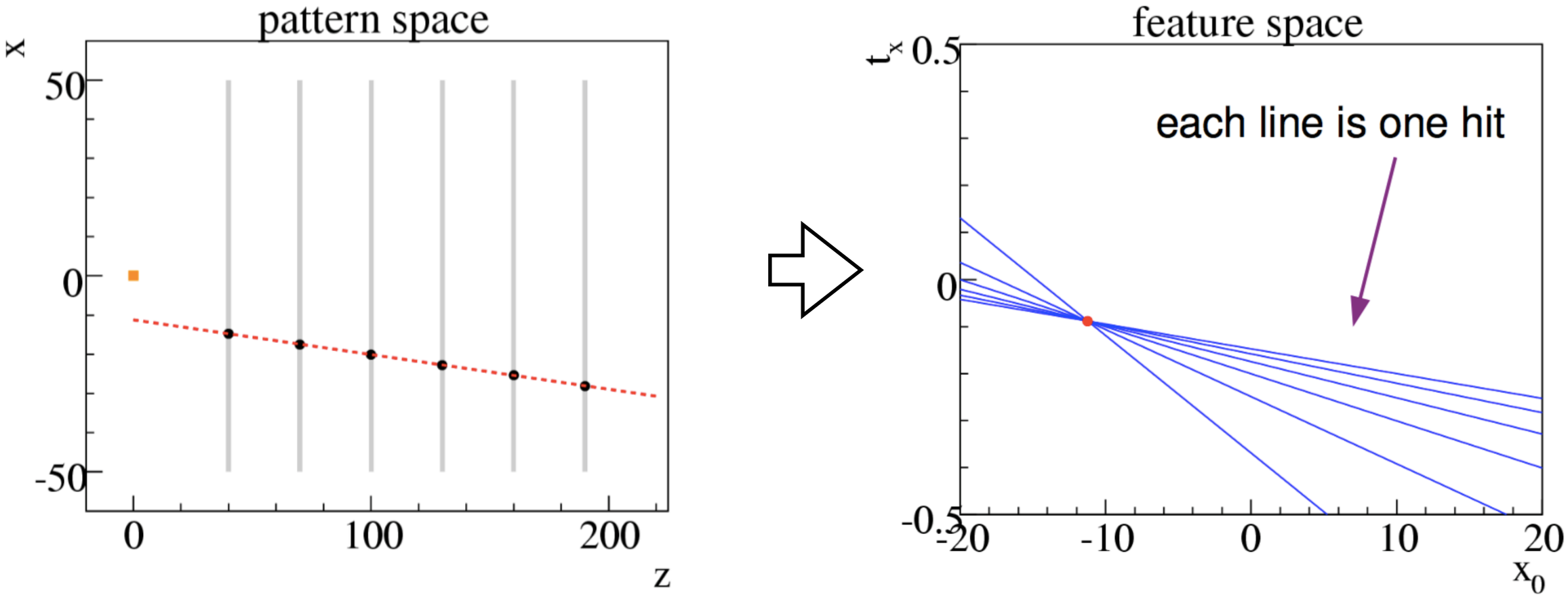
Conformal mapping and Hough transform

Conformal mapping and Hough transform

〉 The idea is to transform your hit information into a parameter space, where your groups of hits are visible

〉 You need a transformation for it which assumes a track model

Conformal mapping and Hough transform

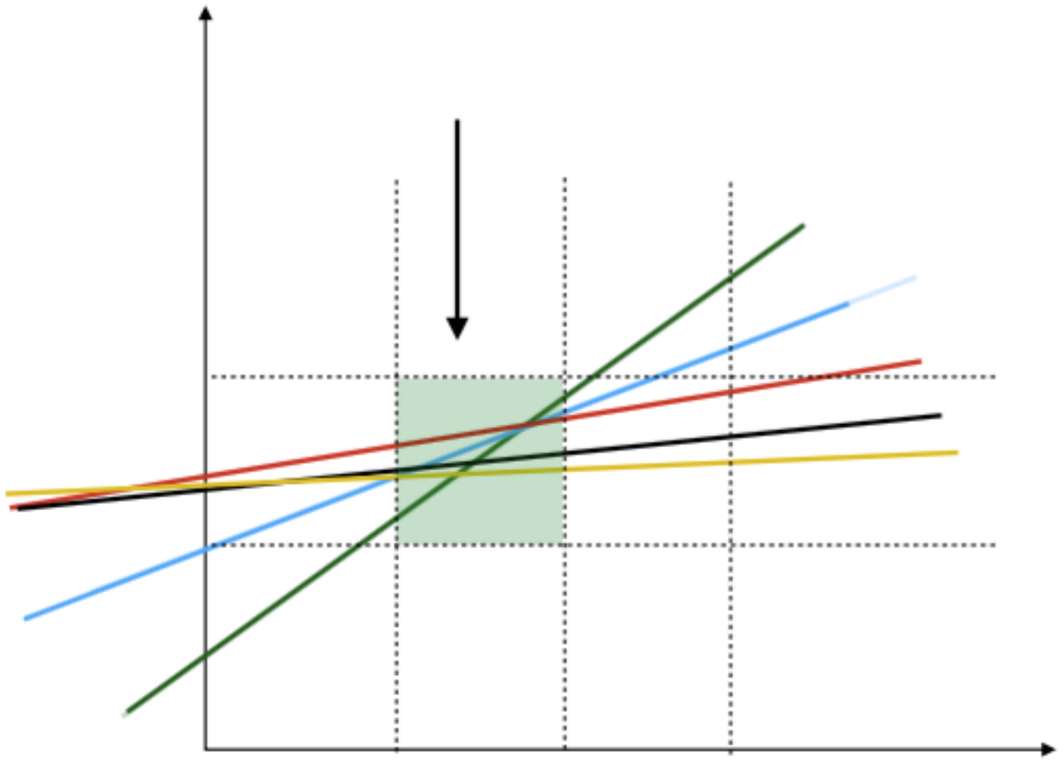


Conformal mapping and Hough transform

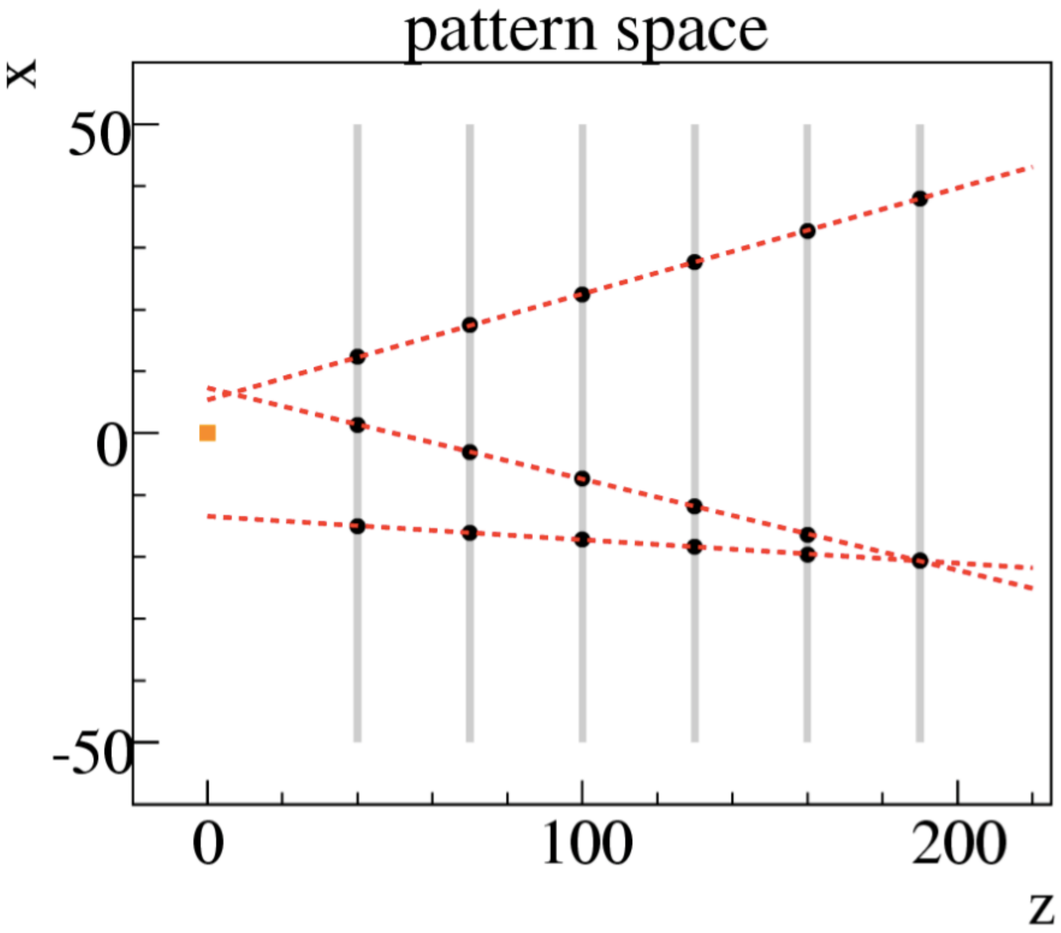
In real applications: finite resolution due to

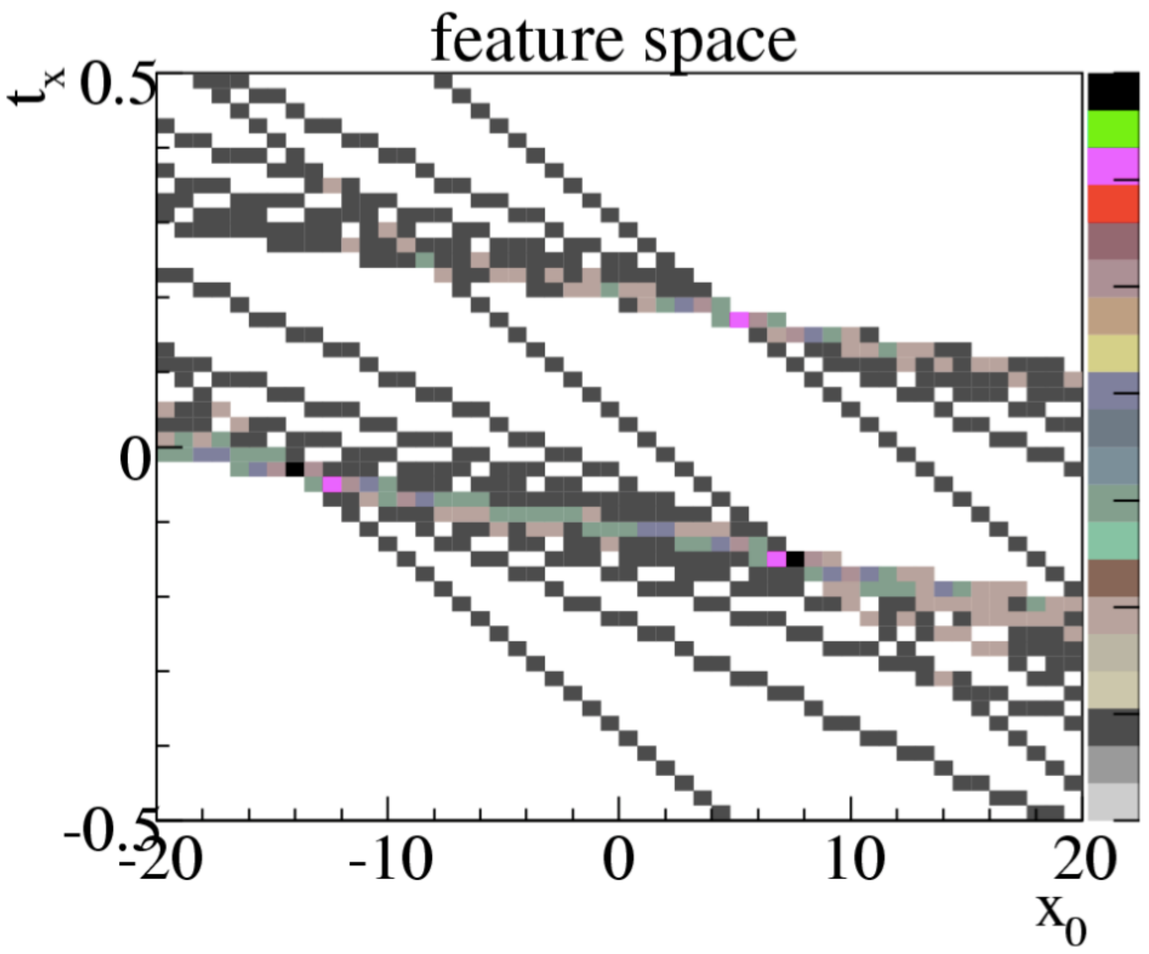
〉 material interaction

〉 inhomogeneous magnetic field



Conformal mapping and Hough transform





Conformal mapping and Hough transform

〉 Works also in higher dimension feature space (e.g. add momentum), but finding maxima becomes more complicated (and time consuming)

〉 Becomes difficult with high occupancy and project noise

Artificial neural network techniques

Artificial neural network techniques

〉 The human brain is particularly skilled in recognizing patterns.

〉 It is capable of analyzing patterns in a global manner; it is self-organizing, adaptive and fault-tolerant.

〉 The human brain is the massively parallel processing of information

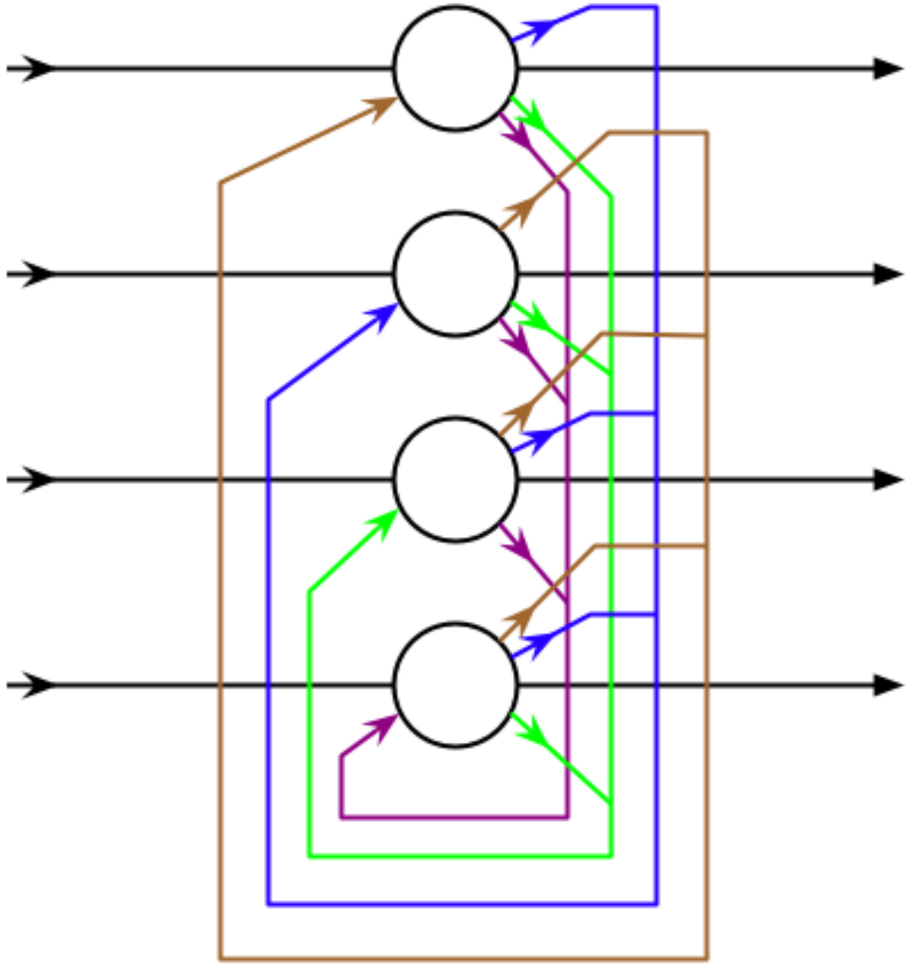
Artificial neural network techniques

ANN algorithms look for global patterns using local (neighbour) rules:

〉 build a network of neurons, each with activation state S

〉 update neuron state based on state of connected neurons

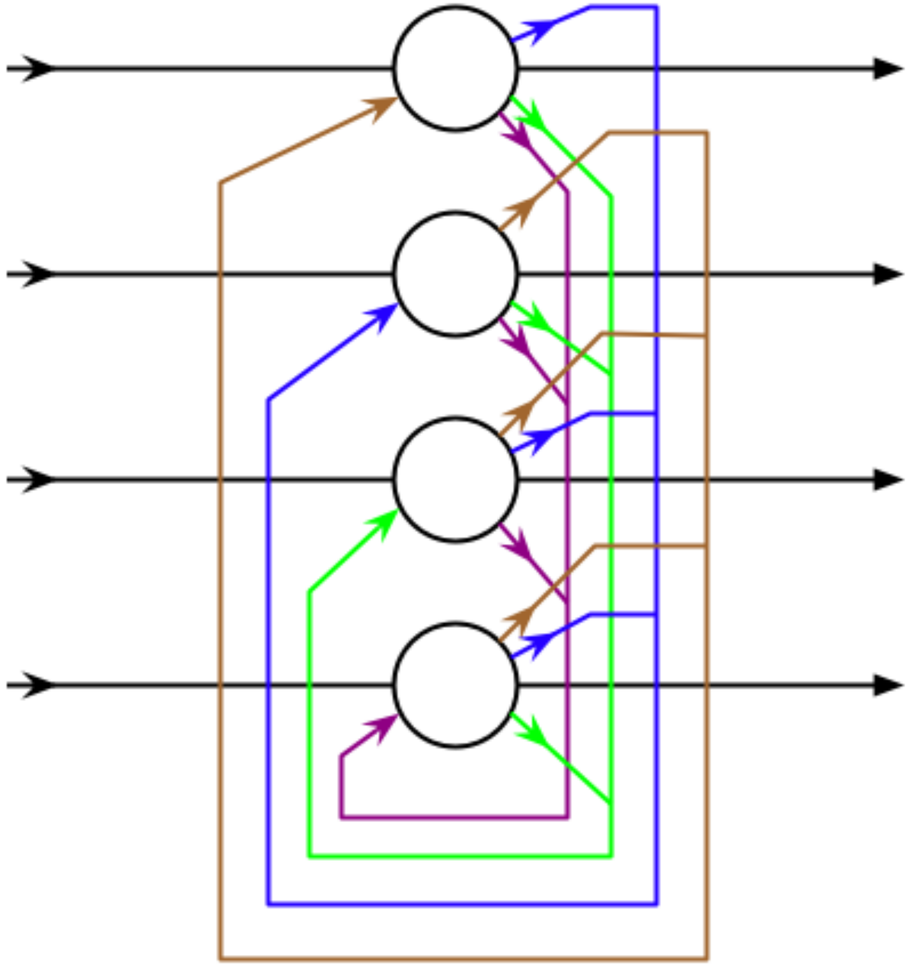
〉 iterate until things converge



The Hopfield neuron

〉 Activation S, can only be either active (1) or inactive (0)

〉 The update rule: $S_i = \Theta\left(\sum_j (w_{ij} S_j - s_i) \right)$



The Hopfield neuron

〉 The update rule: $S_i = \Theta\left(\sum_j (w_{ij} S_j - s_i) \right)$

〉 Such interactions characterize a system with an energy function

$$E = -\frac{1}{2} \left(\sum_{ij} w_{ij} S_i S_j - 2 \sum_i s_i S_i \right)$$

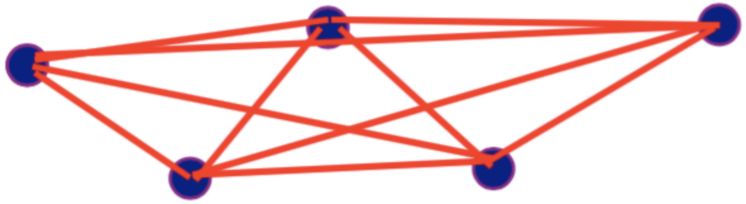
that the interaction leads to a final state that corresponds to the minimum of the energy function

Denny-Peterson neural net

Neuron has two different states:

〉 S(ij) = 1 if two hits belong to same track

〉 S(ij) = 0 if two hits belong to different tracks



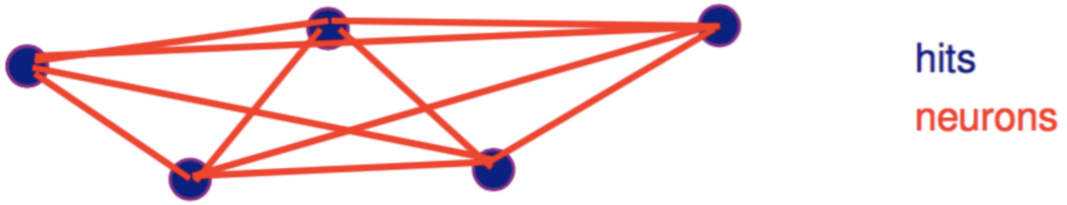
Denny-Peterson neural net

An 'energy' function depends on things like:

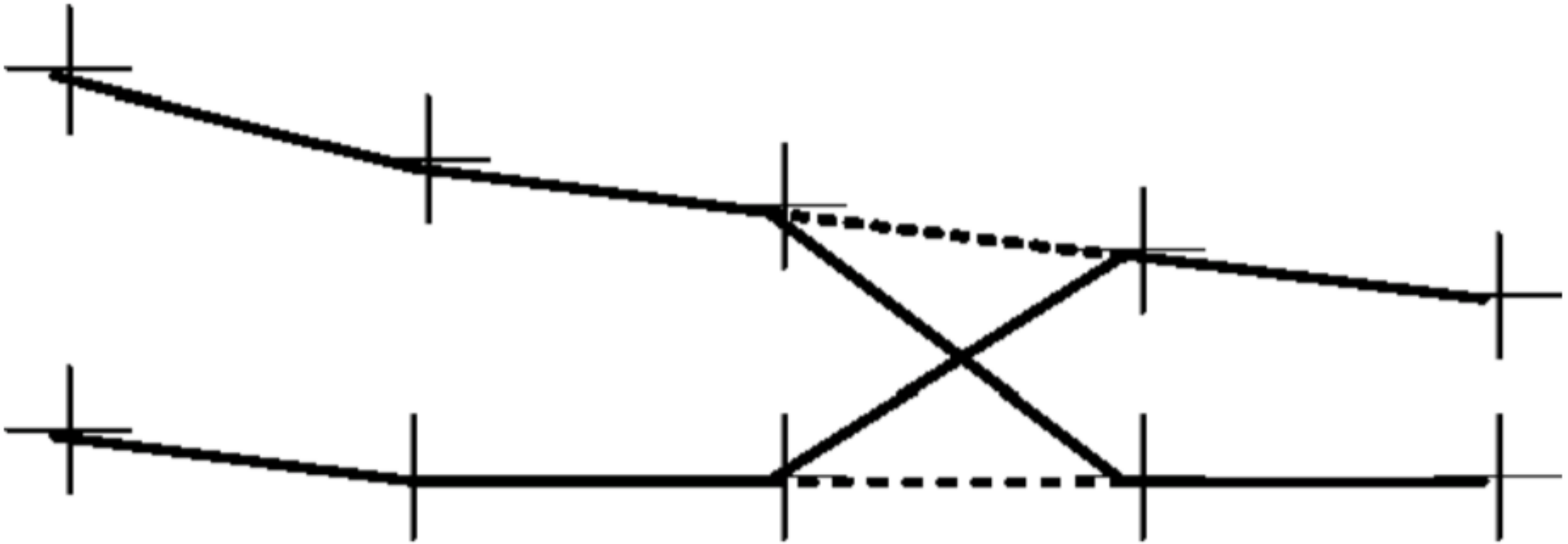
〉 angle between connected neurons: in true tracks neurons parallel

〉 how many neurons: number of neurons ~ number of hits

Track finding becomes 'minimizing energy function'



Denny-Peterson neural net



Denny-Peterson neural net

$$E = -\frac{1}{2}\sum \frac{-\cos^m \theta_{ijk}}{d_{ij}+d_{jk}} S_{ij}S_{jk} + \frac{\alpha}{2}\left(\sum_{l\neq j} S_{ij}S_{il} + \sum_{k\neq i} S_{ij}S_{kj}\right) + \frac{\delta}{2}\left(\sum S_{kl} - N\right)^2$$

'cost function':

- θ_{ijl}: angle between neurons ij and jl

d_{ij}: length of neuron ij

penalty function against bifurcations

penalty function to balance number of active neurons against number of hits

Minimize energy with respect to all possible combinations of neuron states

Denny-Peterson neural net

Difficulties:

〉 Alpha, delta and m are adjustable parameters

〉 With discrete states, minimization not very stable. Therefore, define continuous states and an update function

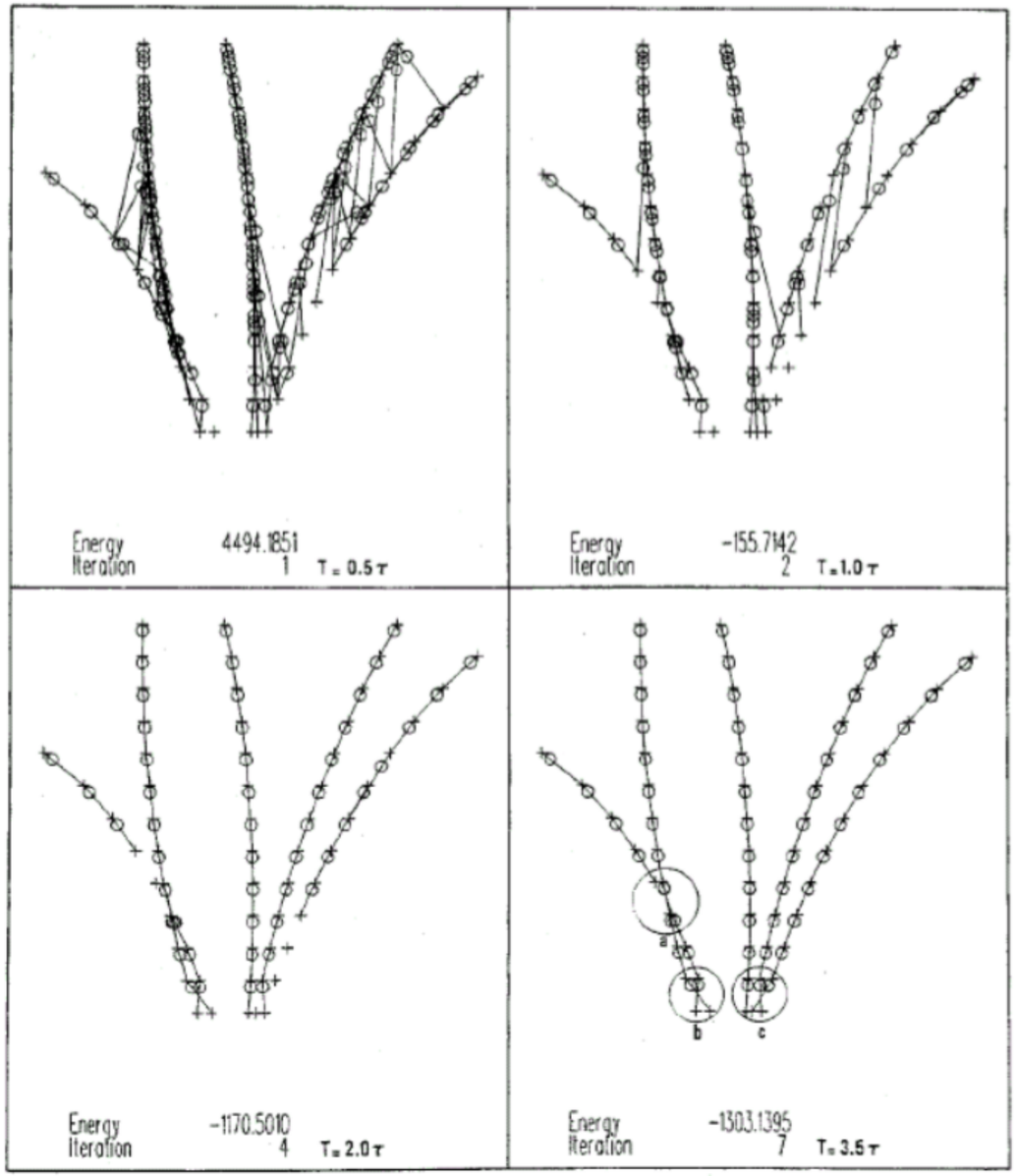
$$v_{ij} = \frac{1}{2}\left(1 + \tanh\left(-\frac{\partial E}{\partial v_{ij}}\frac{1}{T}\right)\right)$$

Denny-Peterson neural net

The algorithm:

1. Create neurons, initialize with some state value. Usually a cutoff on d(ij) is used to limit number of neurons.
2. Calculate the new states for all neurons using the previous equation.
3. Iterate until things have converged, eventually reducing temperature between iterations.

Denny-Peterson neural net



Cellular automaton

Like Denby-Peterson neural net, but simpler

Start again by creating neurons ij:

〉 connect only hits on different detector layers

〉 each neuron has integer-valued state S(ij), initialized at 1

Cellular automaton

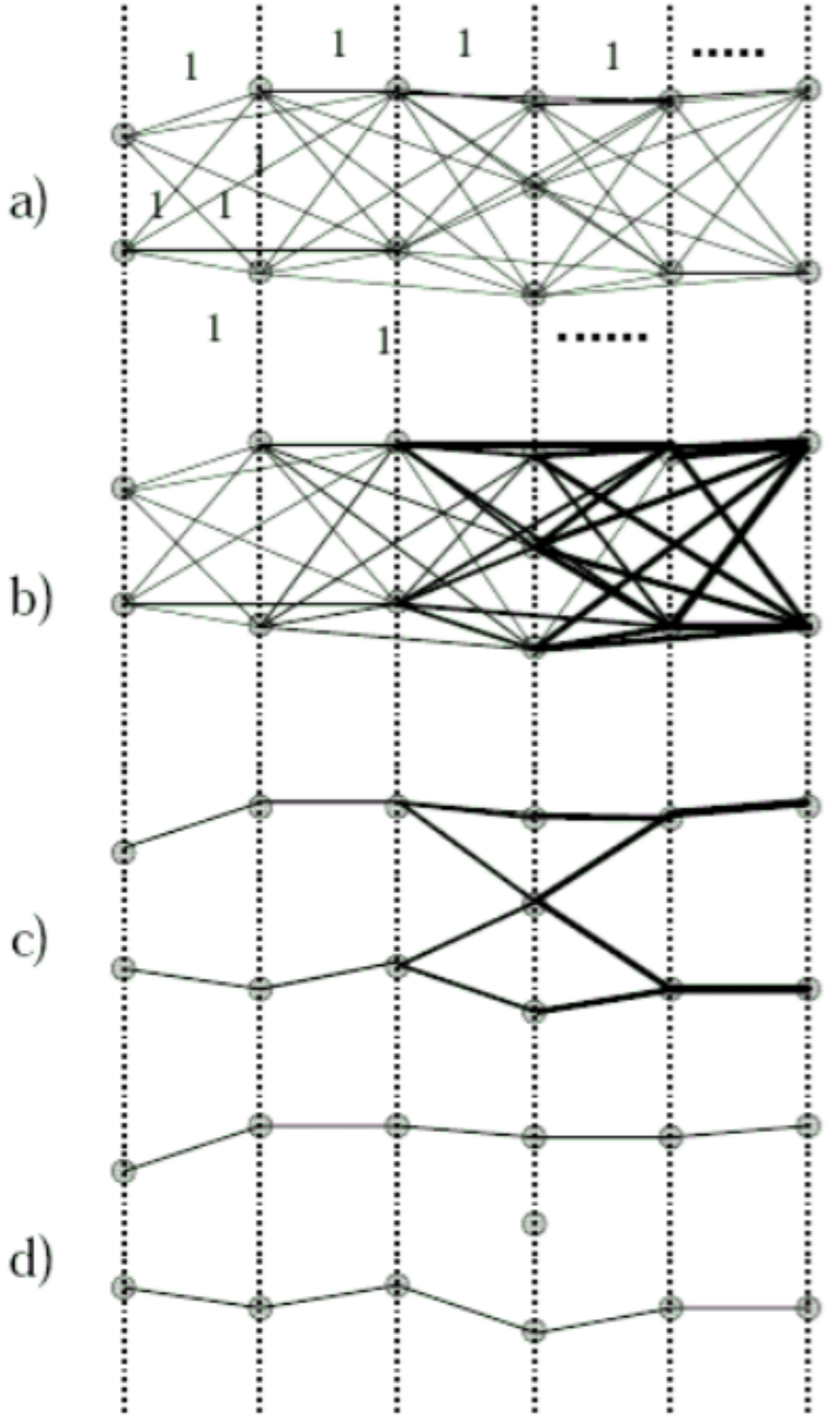
〉 Make a choice about which neuron combination could belong to same track, for example, just by angle

〉 Evolution: update all states simultaneously by looking at neighbours in layer before it

$$S_{ij} = \max\{S_{ki} | \theta_{kij} < \theta^{\max}\} + 1$$

〉 Iterate until all cells stable

〉 Select tracks by starting at highest state value in the network



initialization

end of evolution:
state value indicated by line thickness

selection of longest tracks

more selection to remove overlapping tracks with same length

Elastic arms and deformable templates

〉 Denny-Peterson neural net finds something straight : no difference between track bended in magnetic field and track with random scatterings

〉 Hard to extend to situation with magnetic field

〉 Limitations are overcome by the elastic arms algorithm,

which works with deformable track templates

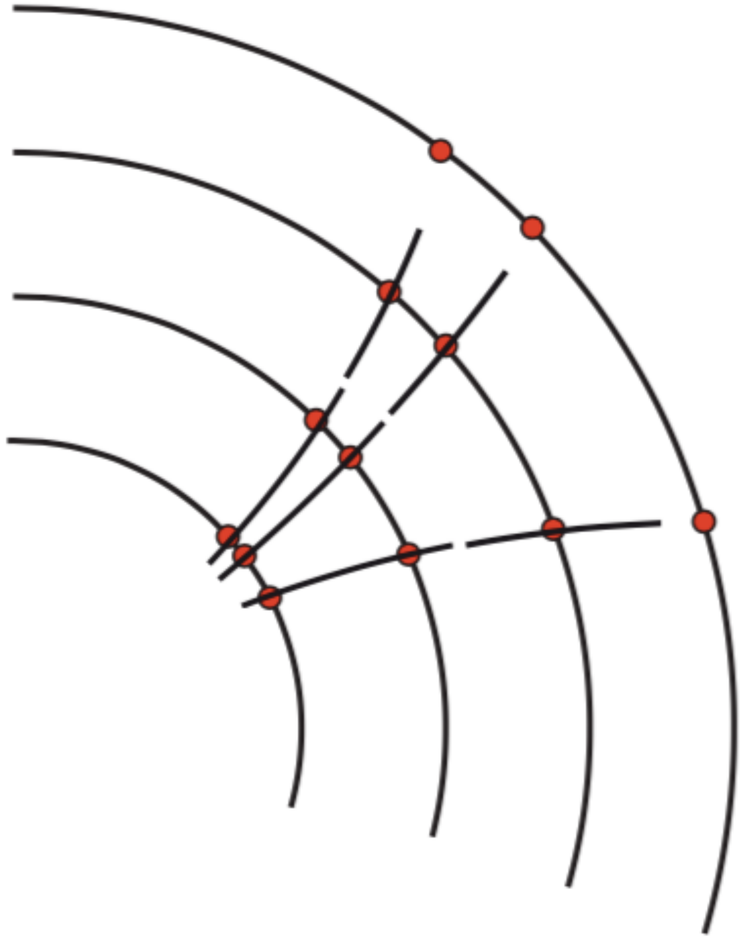
Track Finding

Local methods

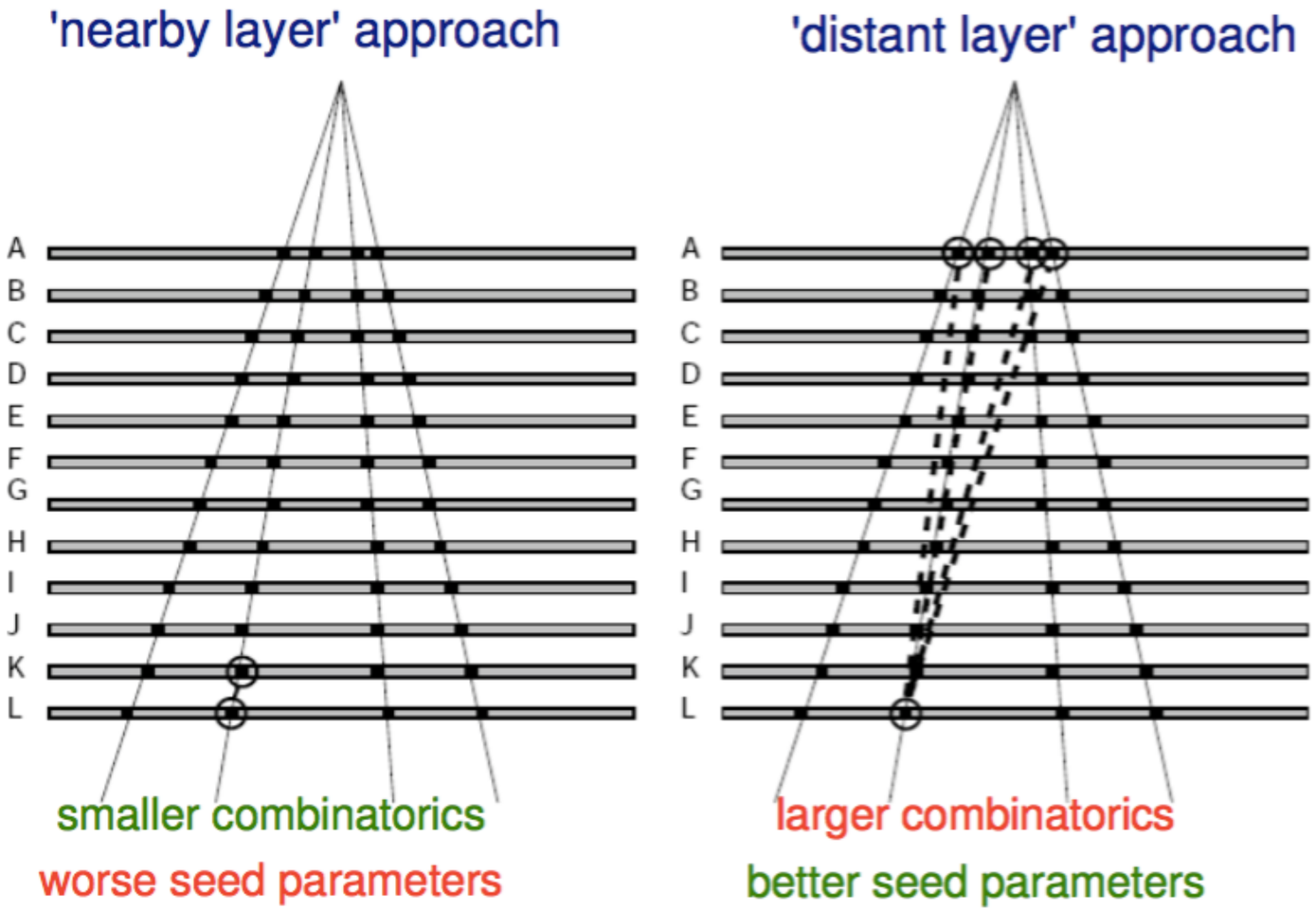
Local methods

Start of many track finding algorithms is the building of track seeds:

- groups of 2 or 3 measurements that are compatible with a crude track hypothesis
- seeds are used to build roads to find track candidates
- usually, seeds are created in region with lowest occupancy



Local methods



Naive track following

Naive track following

1. Starting from a seed, the trajectory is extrapolated to the detector part where the next hit is expected.
2. If a suitable hit is found, it is appended to the track candidate.
3. Where several hits are possible naive track following selects the one closest to the extrapolated trajectory.
4. This procedure is continued until the end of the tracking area is reached, or no further suitable hit can be found.

Naive track following

Problems:

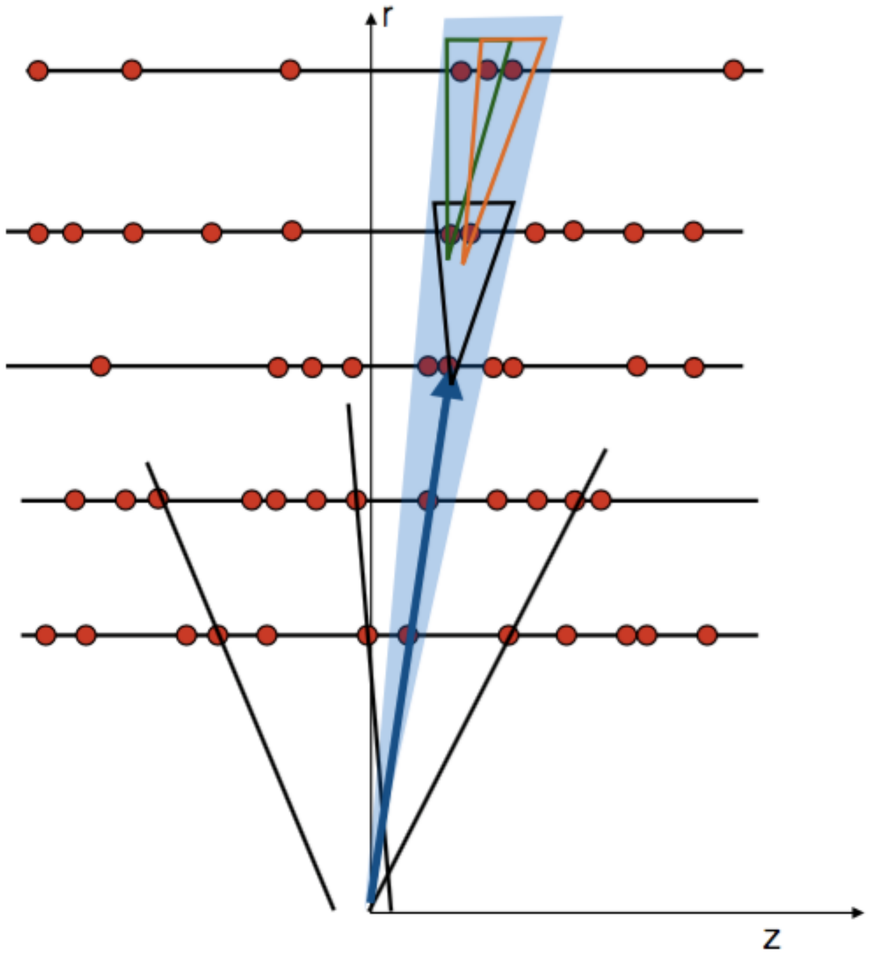
〉 Detector inefficiency may lead to track being rejected for wrong reason

〉 Wrong hit may be closer than correct hit

The combinatorial filter

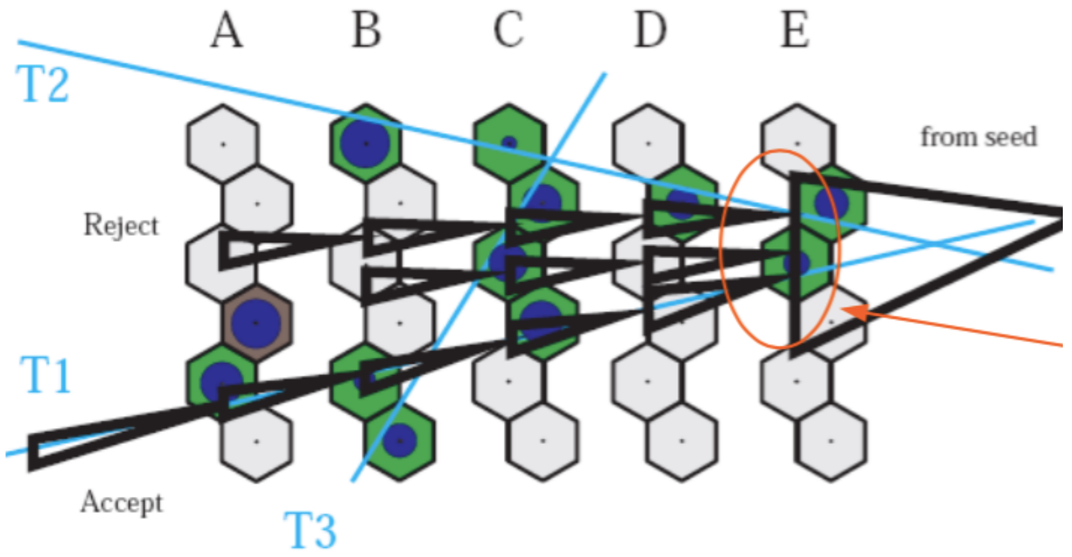
The combinatorial filter

There may not always be one obvious path to be followed



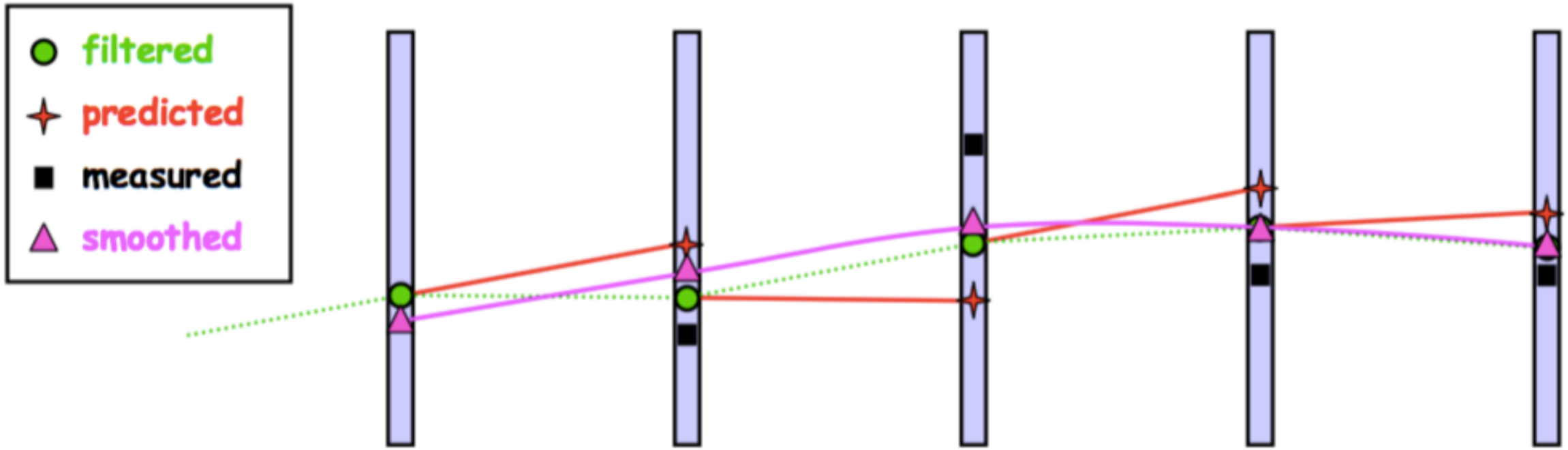
The combinatorial filter

1. Split seed if more than one hit compatible
2. Follow both seeds, reject seeds with two many missing hits
3. After all layers processed, select between overlapping tracks (number of hits/holes, track chi-square etc.)



Kalman Filter

Kalman Filter



Kalman Filter

The filtering is nothing but a weighted average of the

new measurement y_n

the prediction y_p

$$y_f = \frac{\frac{1}{\sigma_p^2} y_p + \frac{1}{\sigma_n^2} y_n}{\frac{1}{\sigma_p^2} + \frac{1}{\sigma_n^2}}$$

$$y_f = \frac{\sigma_n^2}{\sigma_p^2 + \sigma_n^2} y_p + \frac{\sigma_p^2}{\sigma_p^2 + \sigma_n^2} y_n$$

Kalman Filter

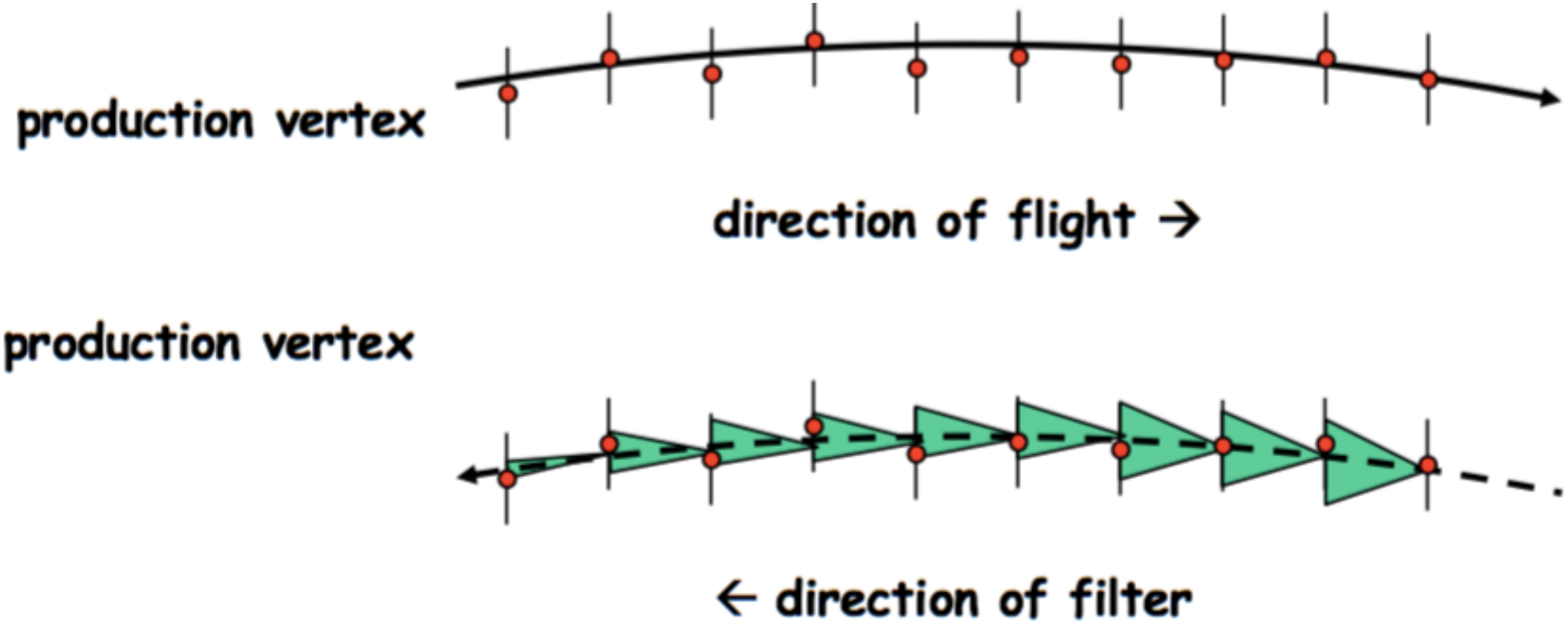
- If the new measurement has a very large error - measurement ignored
- If the prediction has a large error - prediction ignored

Kalman Filter

The advantages of this procedure are:

- is an iterative procedure
- not necessary to invert large matrices
- is a local procedure

Kalman Filter



Conclusion

- Most robust strategies involve more than one track finding algorithm
- There is no universal solution

Thank you for the attention!

Contacts

Mikhail Hushchyn

mikhail91@yandex-team.ru

Andrey Ustyuzhanin

anaderi@yandex-team.ru