Institute for High Energy Physics

Thoughts about Multi-Vertex Finder Strategy

Thorsten Lux

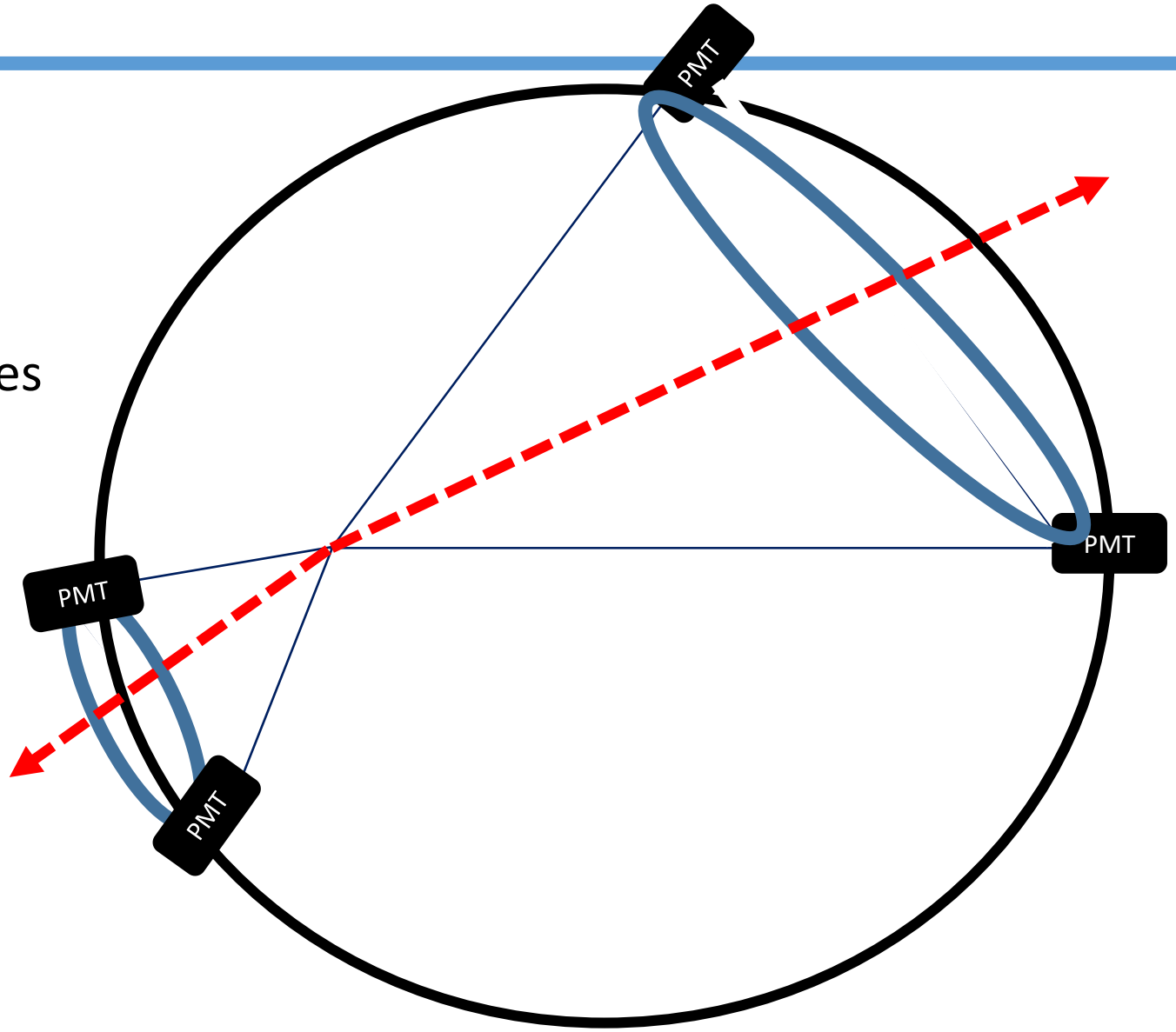
Possible Problem: Light Travel Time

Institute for High Energy Physics

- Each PMT provides:
  - Charge: q
  - Hit time: t
  - Position: x,y,z
- Without knowledge of vertex measured q and t might be misleading
- Problems:
  - Photons from same vertex detected by different PMTs might have very different hit times
  - In the case of two vertices hit times order might be even opposite to vertex times
  - Minor issue: light might be absorbed/scattered on path

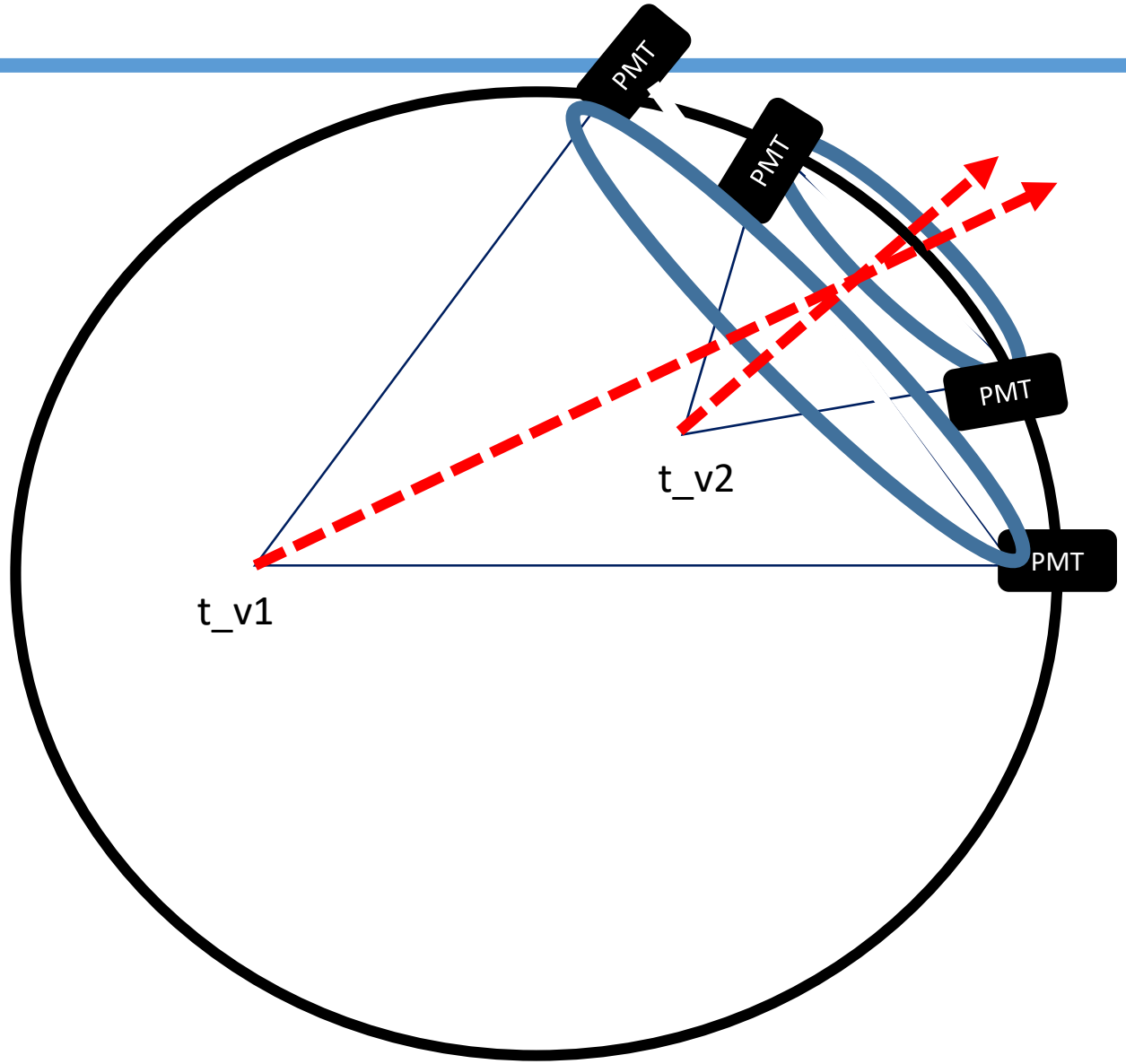
Example 1

- Single vertex
- 2 tracks
- 4 PMTs with different hit times



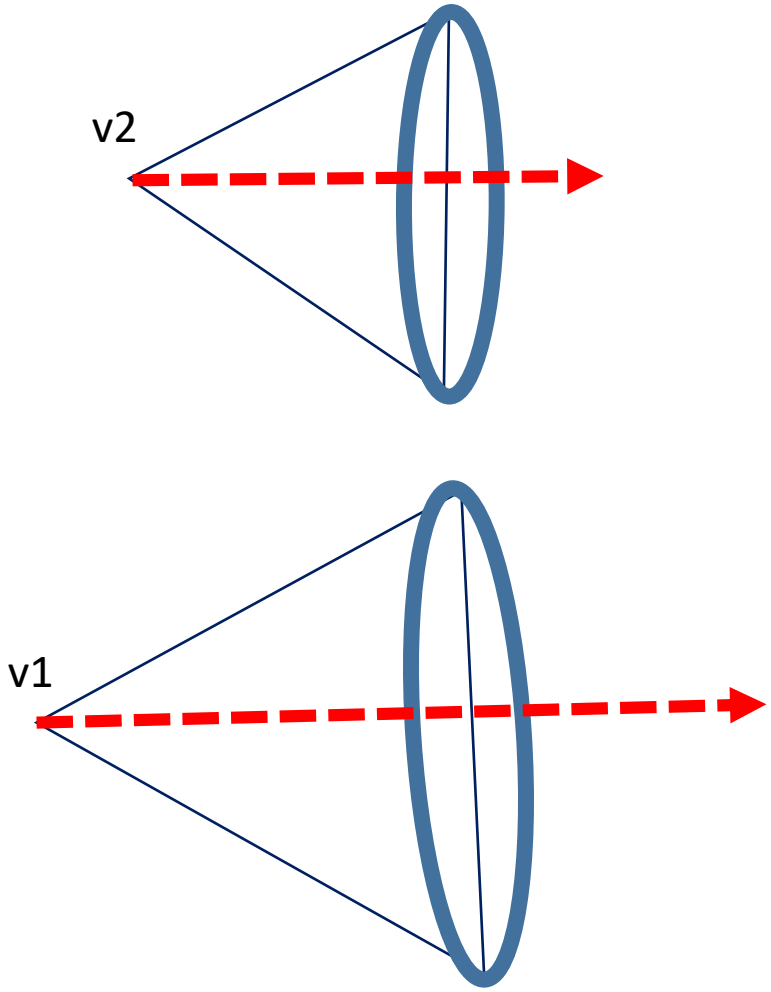
Example 2

- 2 vertices
- t_v2 > t_v1
- 2 tracks
- Hit times measured by PMTs might be inverted giving the impression that t_v2=<t_v1



Physicist vs brute force ML

- Physicist:
  - Sees 2 rings of different diameter
  - Knows opening angle
  - Concludes v2 is close to wall than v1
- Machine Learning:
  - “Sees” list of q,t,x,y,z
  - Does not know concept of ring
  - Does not know opening angle is the same for all rings
  - Has to learn everything …
  - Problem: huge number of possibilities even with 2 vertices per event and 5 tracks per vertex only

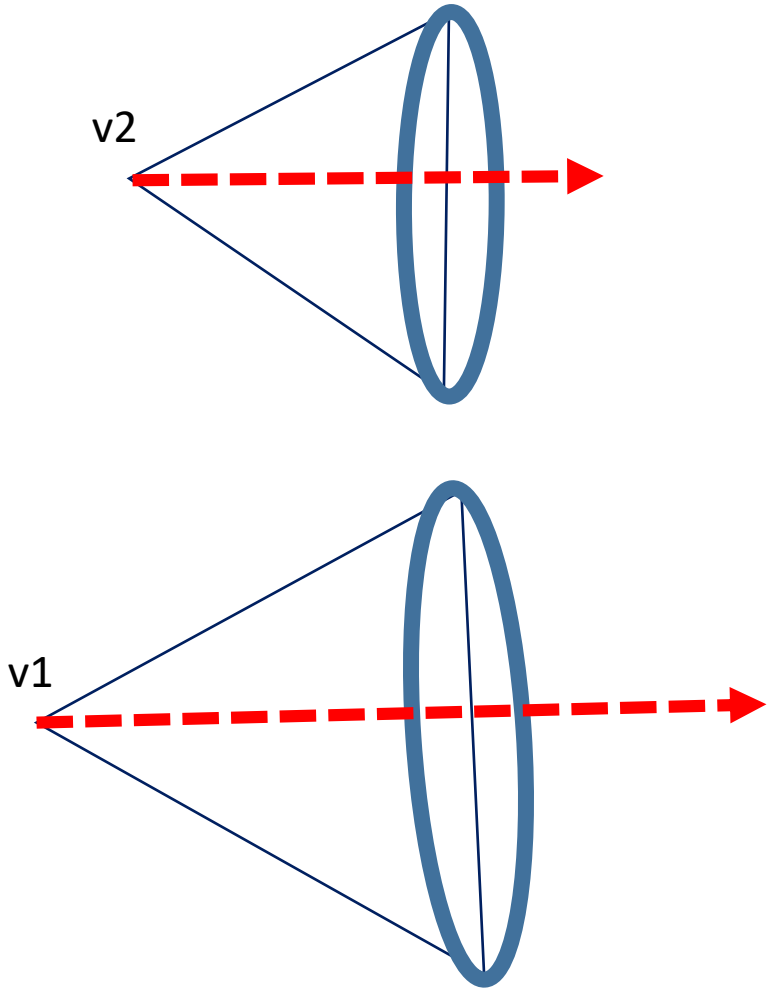


Inefficient?

Worse:

- A lot of training to classify “only” between single vertex and multi vertex events
- Output should be a probability only in this case
- Vertices somehow are reconstructed in a hidden way
- Vertex information (t_v, x_v, y_v, z_v) not provided per event
- Next analysis step which needs the vertex information will have to learn again

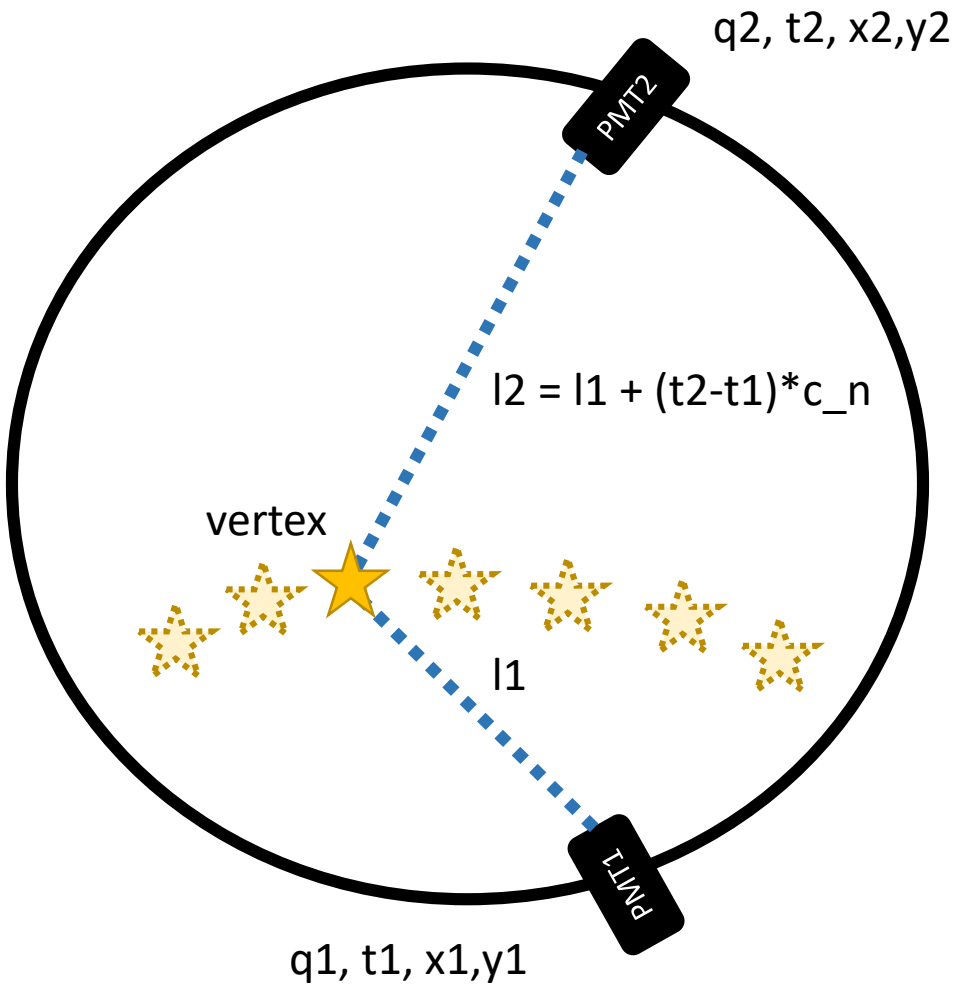
=> Looks horribly inefficient to me! (but I might be wrong)



What we know?

Can we do better?

- Of course, still using ML …
- … but do not use ML to learn what we already know
- **We know:**
  - Photons move on a straight line
  - Cherenkov light is emitted in a cone of 44 degrees opening
  - Photons observed in two different PMTs can only come from the same vertex if the vertex lays on well defined distances from both PMTs (but l1 not fixed)



Option 1: Track Vertices known

Institute for High Energy Physics

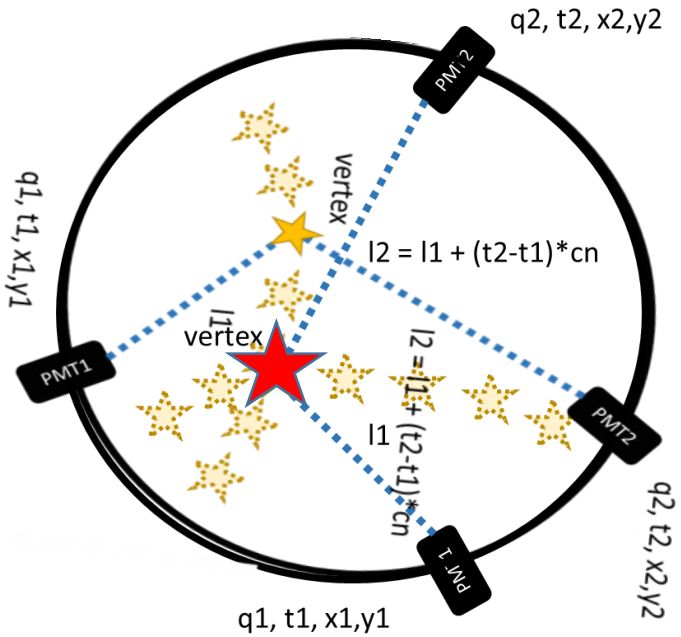
- Divide the job in subtasks e.g.:
  - Classify all hits in Cherenkov hits in real hits and noise hits
  - Use real hits to find rings and separate them
  - For each ring/track find a proto-vertex (x,y,z,t)
  - **Cluster proto-vertices and check if more than one vertex**
  - … continue with higher level steps
- ML could be used for all subtasks
- Could test cluster idea easily:
  - Take MC true vertex position and time
  - For each track smear the vertex position and time
  - Then use ML to see if all come from same vertex

SK:
- Track resolution: e=34 cm, mu=25 cm
- Time resolution: 2 ns

IWCD:
- Expected around 10 cm

Option 2: Reconstruct Vertices using TOF

- Signals in 2 PMTs will give an allowed paraboloid of intersection curves of two spheres
- Every additional PMT ***n+1*** will add ***n*** paraboloids
- Zones in which many paraboloids intersect, a vertex should be
- Using all PMTs might be too much …
- … and perhaps not needed
- All neighbour PMTs with similar hit time will give similar paraboloids => might be enough to use PMTs with certain spatial distance and/or hit time difference



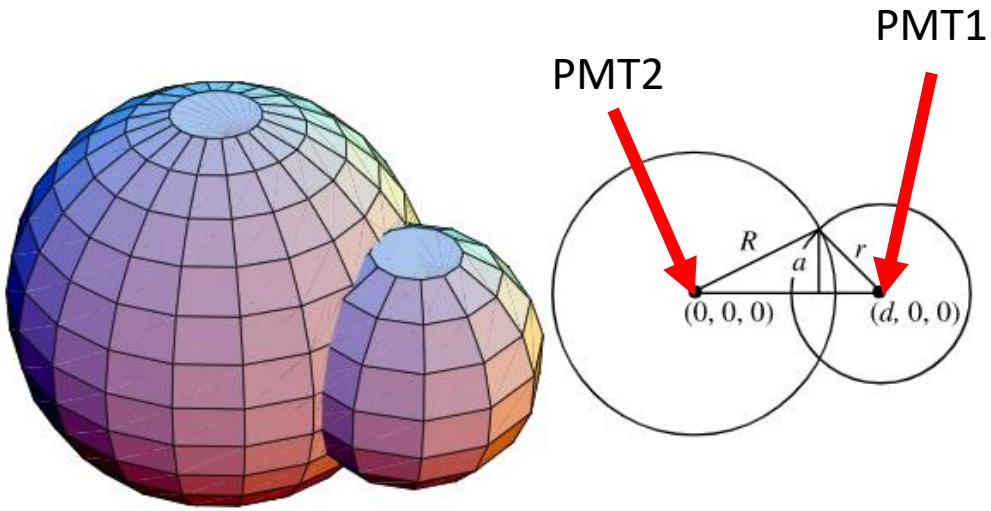
Possible vertex positions:

$$x_v = \frac{d^2 - 2l_1\Delta l - \Delta l^2}{2d}$$

$$y_v = a \cdot \cos\varphi$$

$$z_v = a \cdot \sin\varphi$$

In function of l1 should describe a paraboloid



r= l1, R=l2=l1+dt*cn=l1+dl

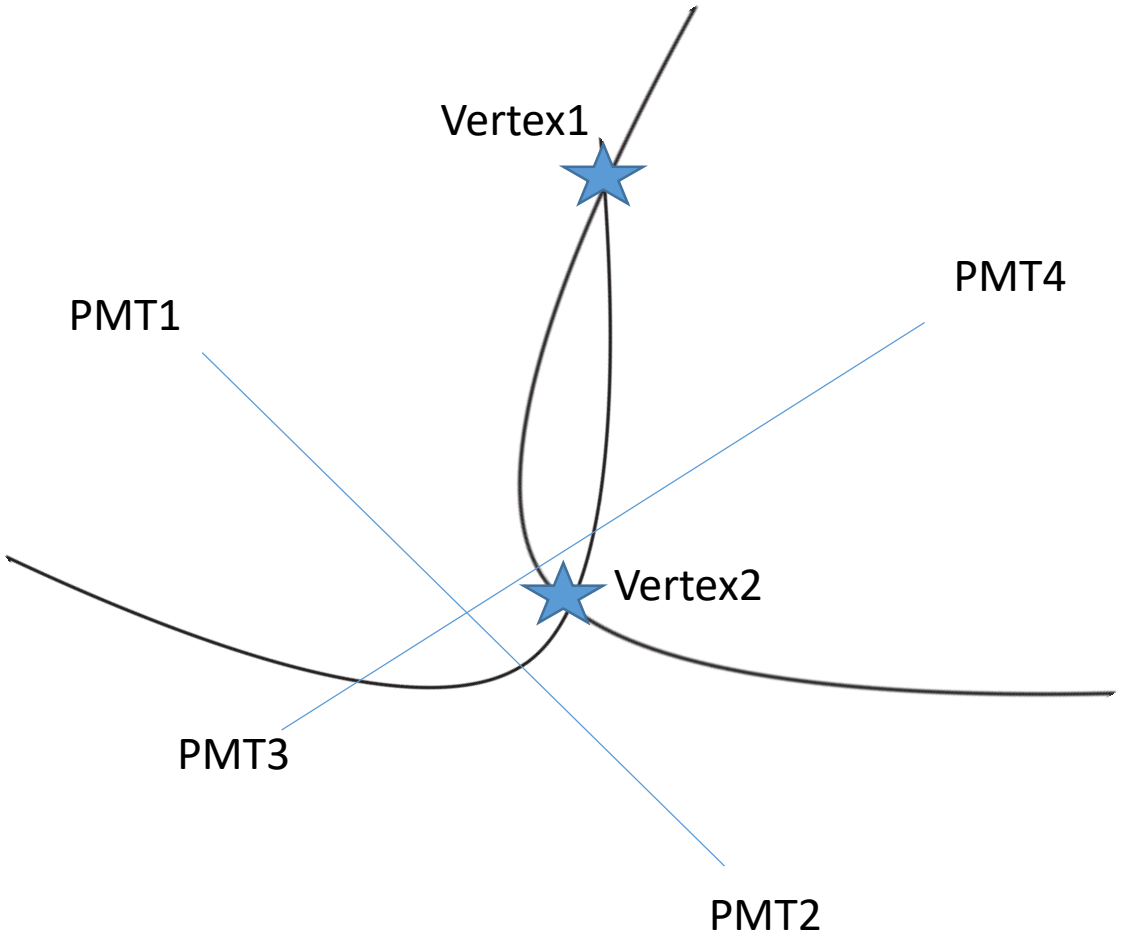
$$a = \frac{1}{2d}\sqrt{(d^2 - \Delta l^2)\left[(2l_1 + \Delta l)^2 - d^2\right]}$$

$$t_v = t_1 - \frac{l_1}{c_n}$$

? hit time

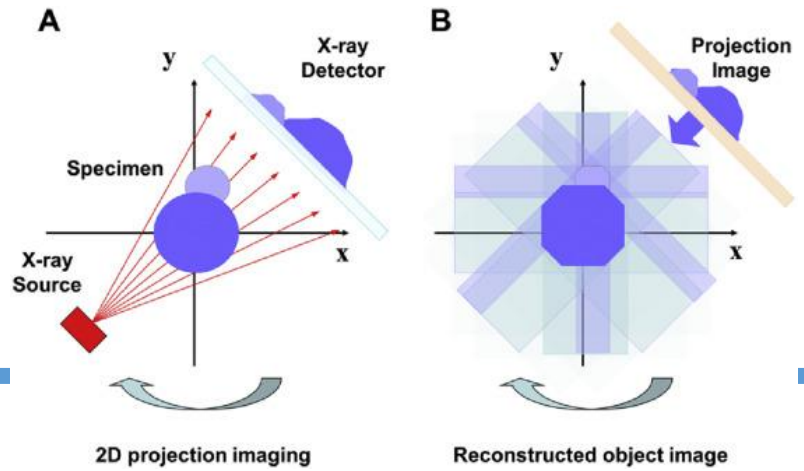
In function of l1 should describe a paraboloid

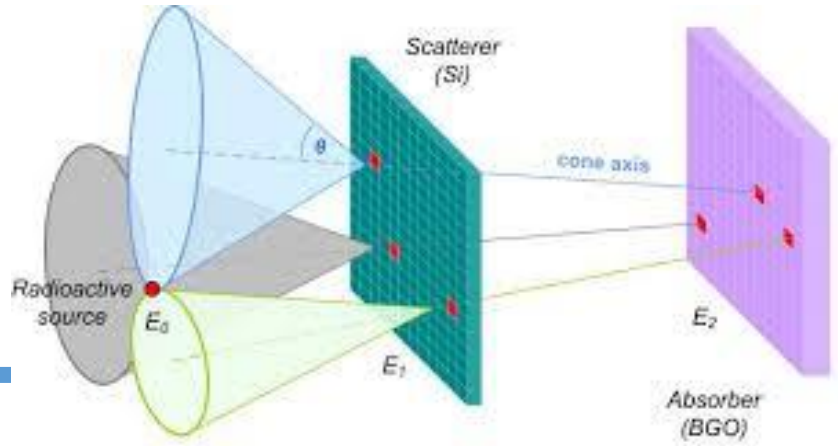
- Under assumption signals in 2 different PMTs come from same vertex, vertex position has to lay on paraboloid surface
- If more pairs of PMTs are used, vertex position limited to intersections
- Using also vertex time, might significantly reduce the possible vertex positions => less combinations needed
- PMTs with similar positions and hit times, will give similar vertex positions => Chose two PMTs from two different mPMTs and calculate the paraboloid surface, check how many PMTs in same mPMT has similar hit times, add weight to paraboloid => significant less combinations
- Could help to find low light rings? Take vertex position for clear rings as in Option 1, remove corresponding PMT hits from list, look for other PMT hits which could come from this vertex …



Medical Imaging Similarities

- Compton Camera: Source position on cone $\cos(\theta) = 1 - m_e c^2 \frac{E_1}{E_2 E_0}$
- Intersection of various cones give source position
- ML used after first step for image reconstruction
- Only 2D image on one position with planar detectors but …
- … 3D image by many merging many 2D images from different views
- Used in CT images
- HK: PMTs placed in 4pi around fiducial volume
- For Pre-vertex finder limited precision could be enough





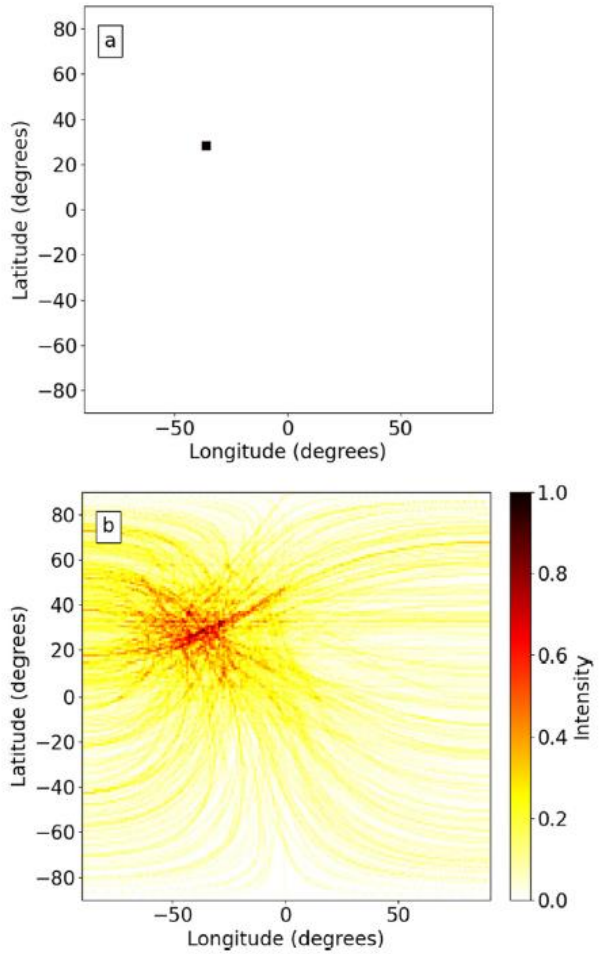


Fig. 3. a: Example of the position of a simulated source, it is the expected output of the neural network. b: Corresponding SBP.

FitQum: Vertex Finding in SK

- Idea came from TOF PET and Compton Camera
- Checked reconstruction in SK
- For vertex reco also TOF was used
- Likelihood defined depending on t and xyz
- Minimized to find vertex position
- Works well for single vertex events (standard in SK)
- Narrow residual time resolution with correct vertex position reconstructed
- Time frames seem also to be used
- **Can we define a variable and use ML to study with ML if there is one or more vertices?**

$$G(\boldsymbol{x}, t) \equiv \sum_{i}^{\text{hit}} \exp(-(T^{i}_{\text{res}}/\sigma)^2/2),$$

$$T^{i}_{\text{res}} \equiv t_i - t - |\boldsymbol{R}^{i}_{\text{PMT}} - \boldsymbol{x}|/c_n.$$

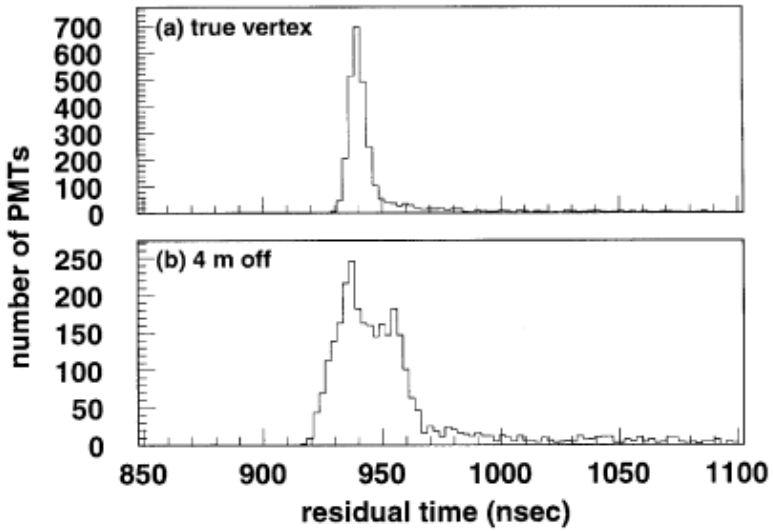


Fig. 1. The residual time distributions after TOF subtraction. The time of flight is calculated with (a) true vertex position and (b) 4 m off from the true vertex position. With the true vertex, the residual time distribution has a sharp peak.

Conclusions

Institute for High Energy Physics

- My doubt is: Is brute force ML the right way?
- Possibly too much resources needed for two little info gain
- Better approach: **Exploit what we know and use ML for what we do not know?**
- Vertex information could be obtained with previous analysis steps
- First test could be simple by using smeared MC information
- One could reconstruct vertices positions with TOF
- Similarly done in SK
- Or let us think about variables we could define to use then ML to decide how many vertices we have in one event!
- Worth to discuss with Ishitsuka-san?