No track left behind

Jonathan Kriewald

IJS

Based on arXiv:2510.00856

Felica 2025

No track left behind: From Silicon hits to LLP reconstruction

Jonathan Kriewald

IJS

Based on arXiv:2510.00856

Felica 2025

No track left behind: From Silicon hits to LLP reconstruction

Jonathan Kriewald

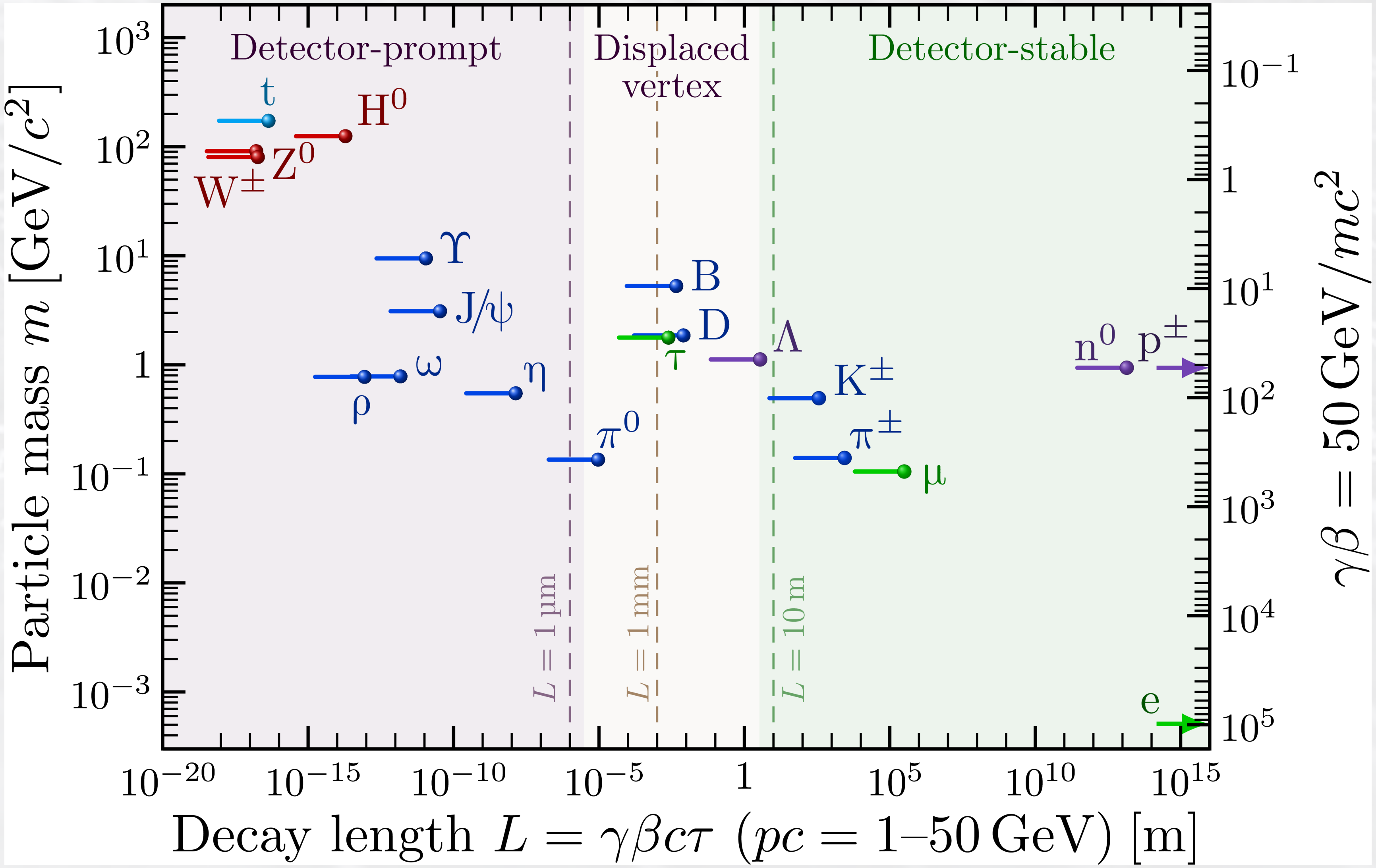
Long-lived Particles

(In the Standard Model)

SM particle **lifetimes** span >30 orders of magnitude

Experiments measure "**detector-stable**" particles

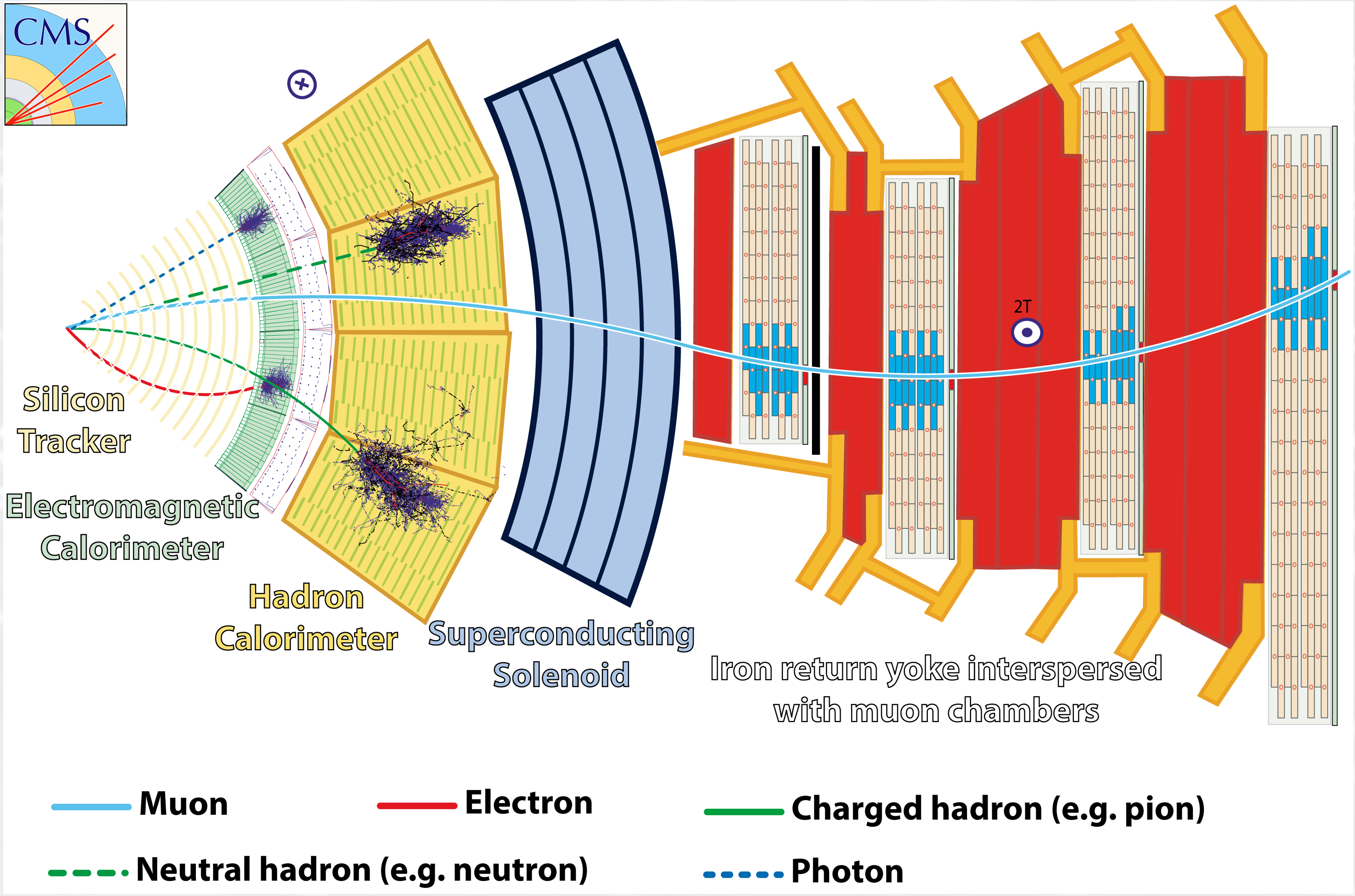
Reconstruct unstable particles from detector-stable **decay products**



Long-lived Particles

(In the Standard Model)

Experiments measure "**detector-stable**" particles



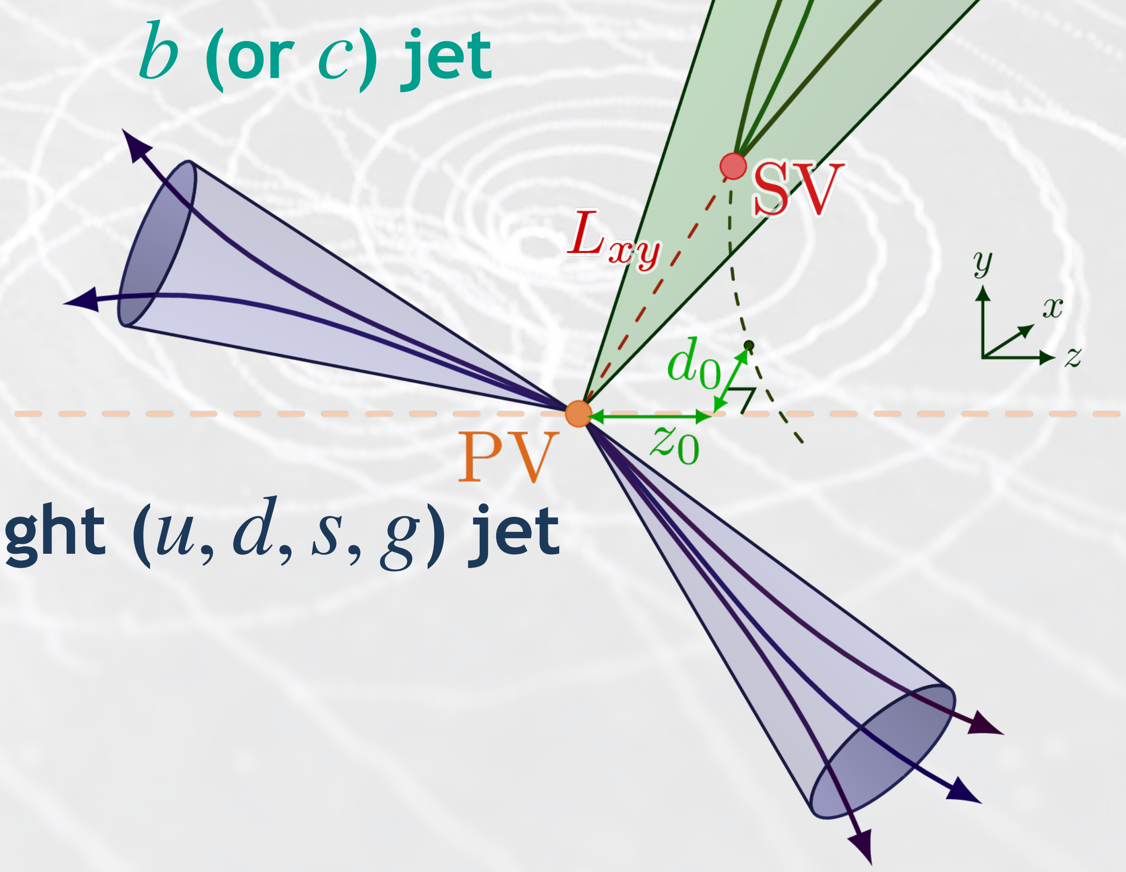
Long-lived Particles

(In the Standard Model)

Reconstruct unstable particles from detector-stable **decay products**

(Displaced) Vertex finding/ fitting **crucial step** in any event **reconstruction**

e.g. jet-tagging:



Long-lived Particles

(In the Standard Model)

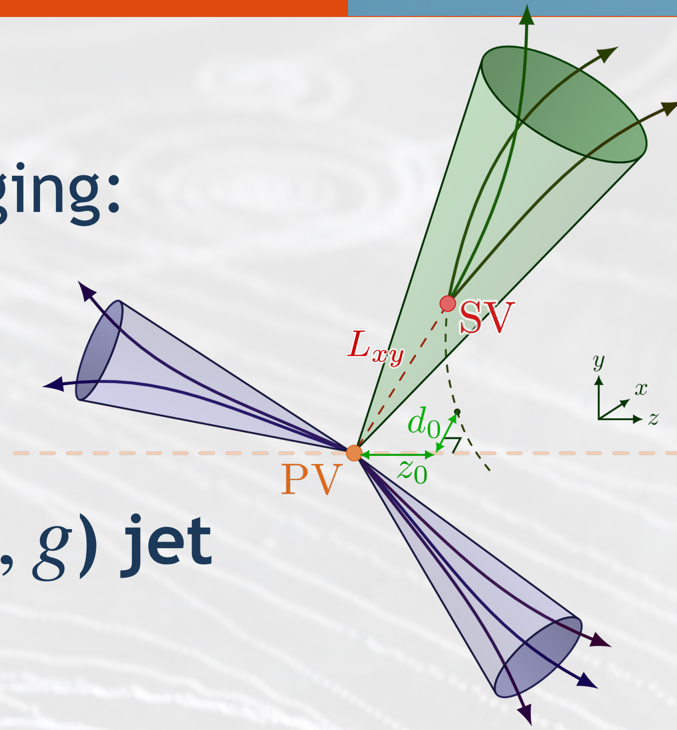
Reconstruct unstable particles from detector-stable **decay products**

(Displaced) Vertex finding/ fitting **crucial step** in any event **reconstruction**

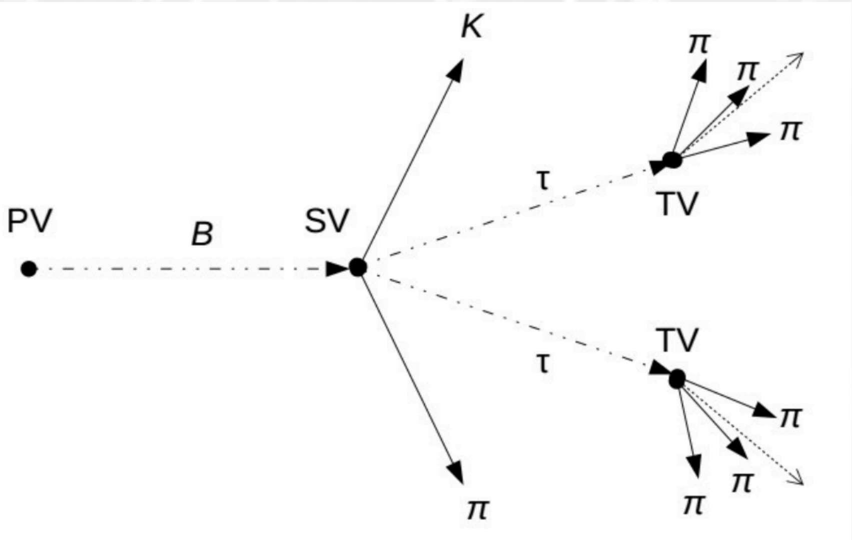
e.g. jet-tagging:

b (or c) jet

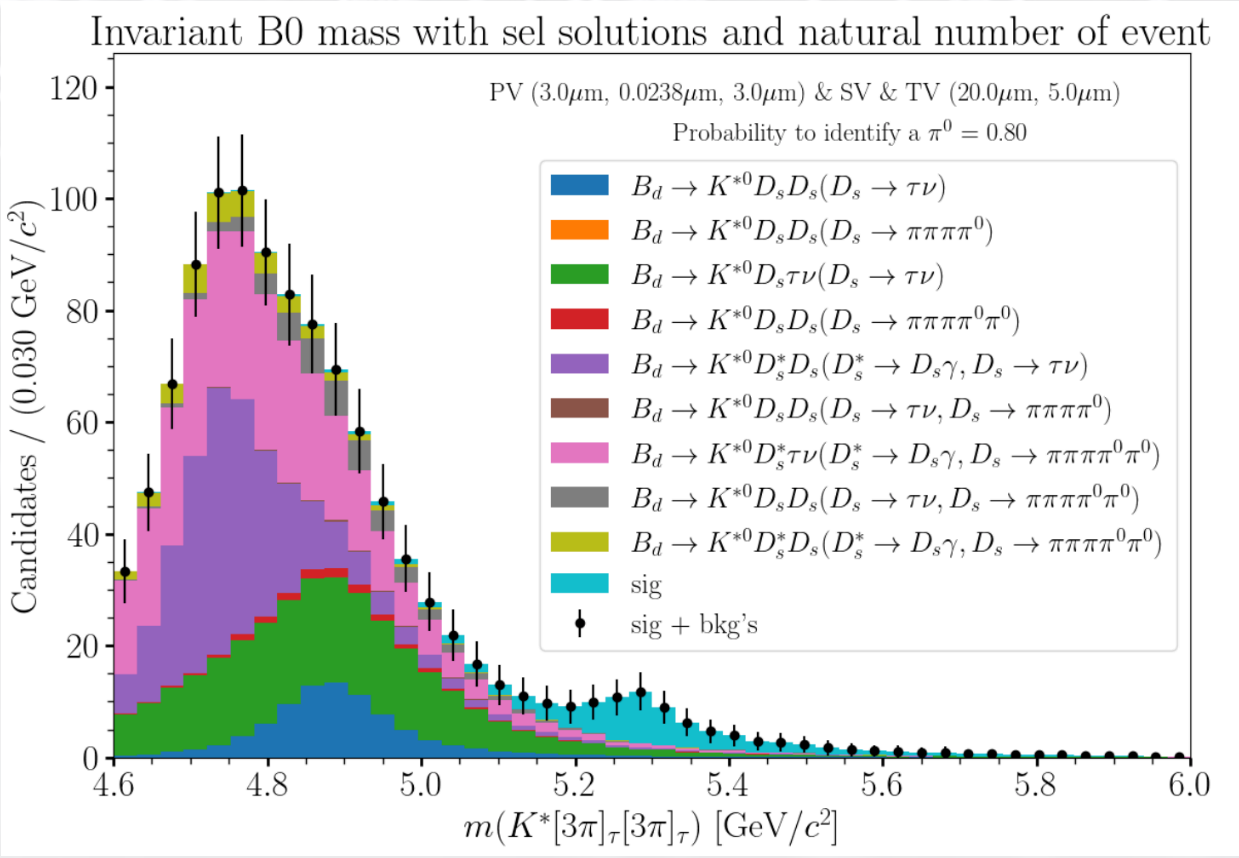
light (u, d, s, g) jet



e.g. $B \to K^*(\to K^+\pi^-)\tau^+\tau^-$



$\Rightarrow$



Tristan Miralles & Stéphane Monteil: FCC Ana note March 2025

Long-lived Particles

(In the Standard Model)

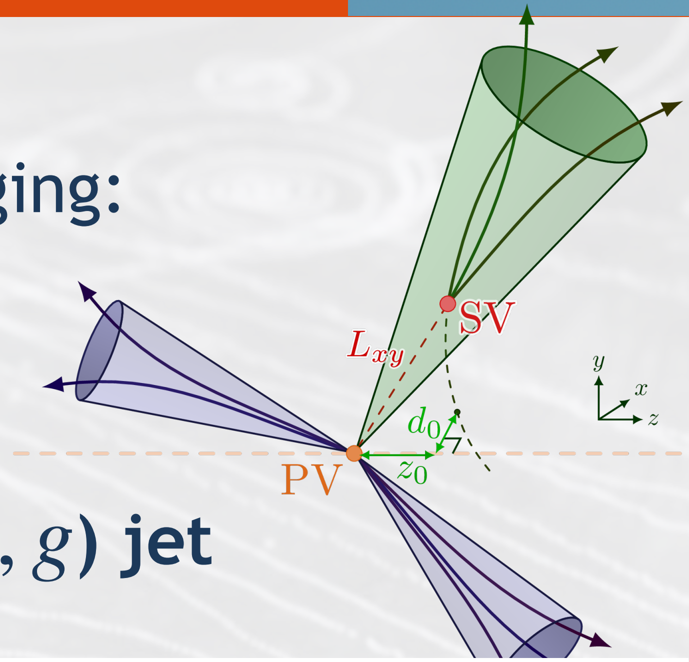
Reconstruct unstable particles from detector-stable **decay products**

(Displaced) Vertex finding/ fitting **crucial step** in any event **reconstruction**

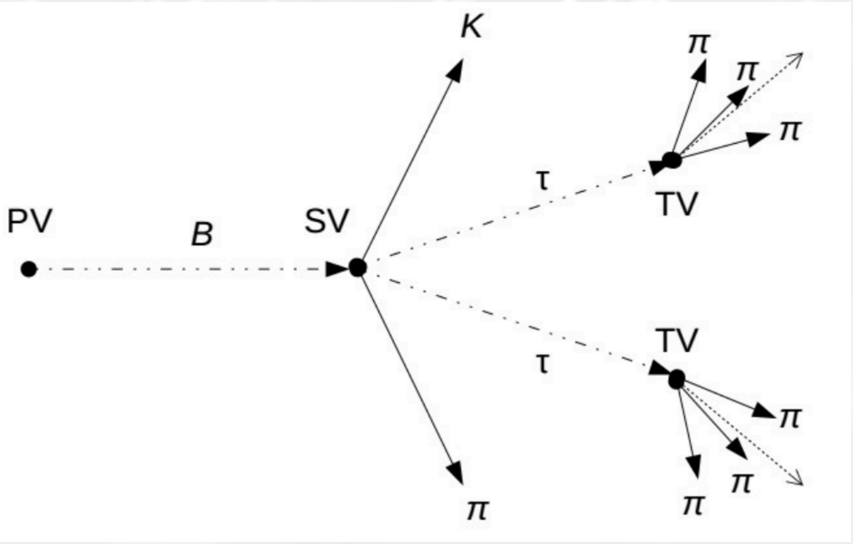
e.g. jet-tagging:

b (or c) jet

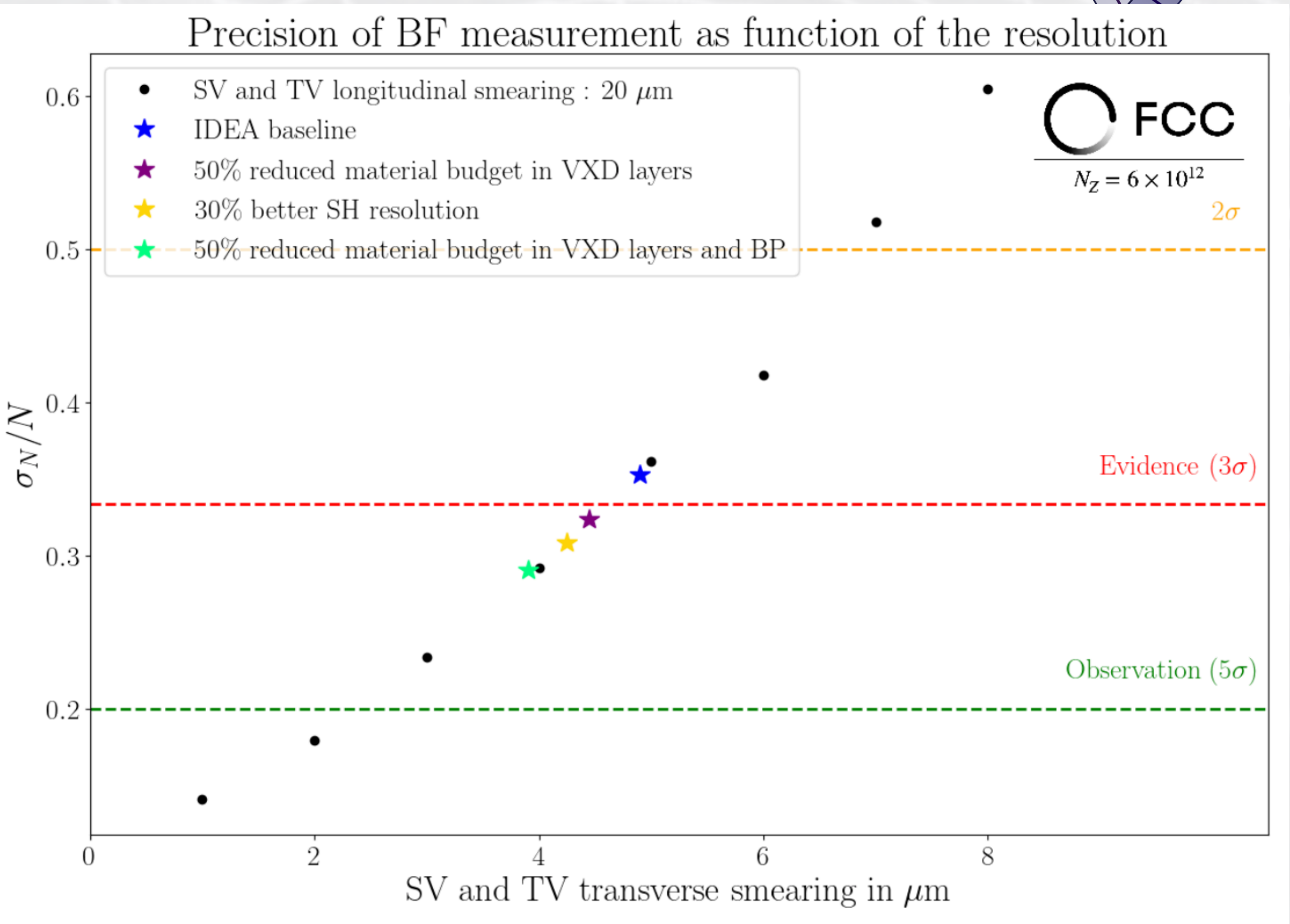
light (u, d, s, g) jet



e.g. $B \to K^*(\to K^+\pi^-)\tau^+\tau^-$



⇒



Tristan Miralles & Stéphane Monteil: FCC Ana note March 2025

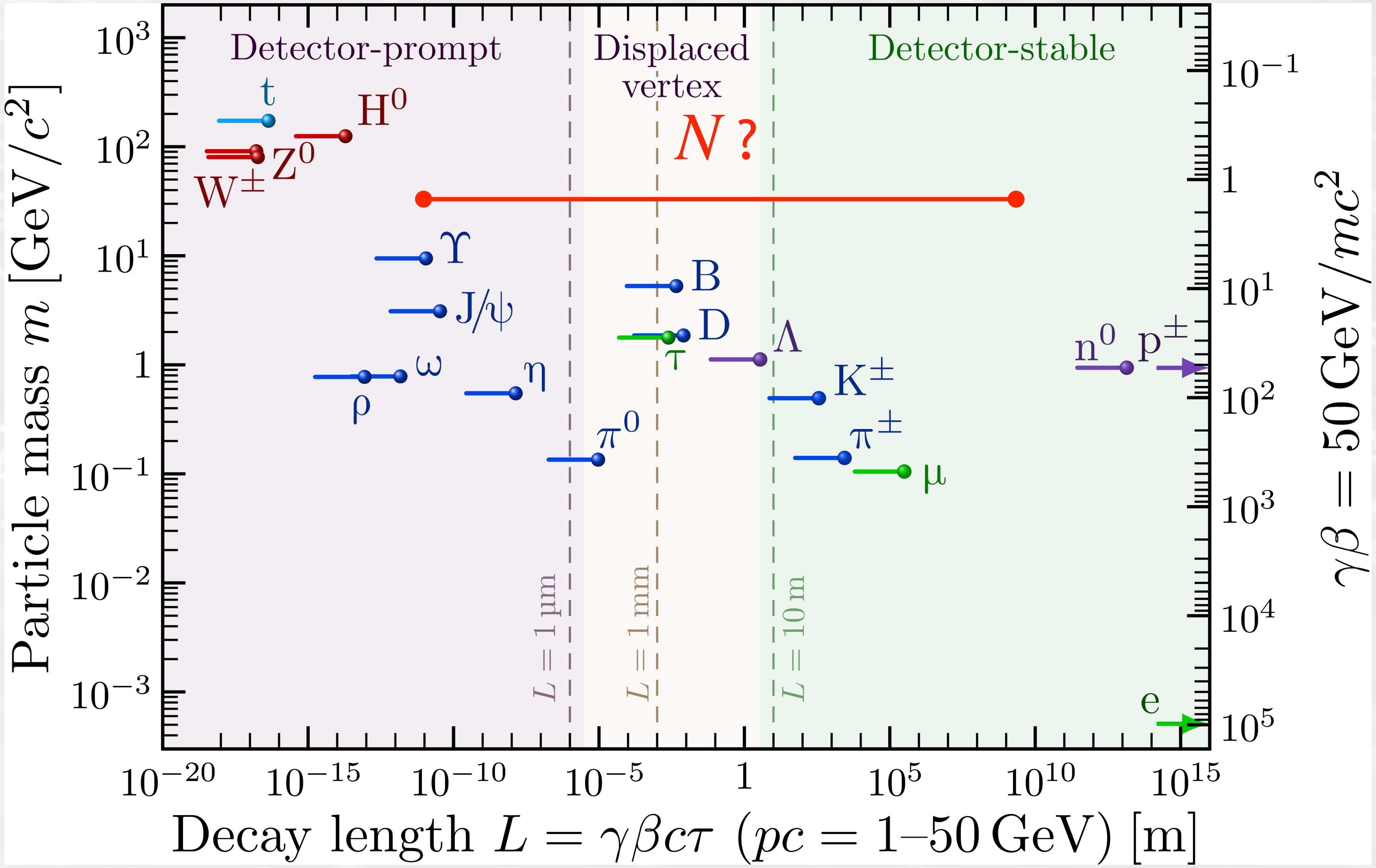
Long-lived Particles

(Beyond the Standard Model)

SM particle **lifetimes** span >30 orders of magnitude

Experiments measure "**detector-stable**" particles

Reconstruct unstable particles from detector-stable **decay products**



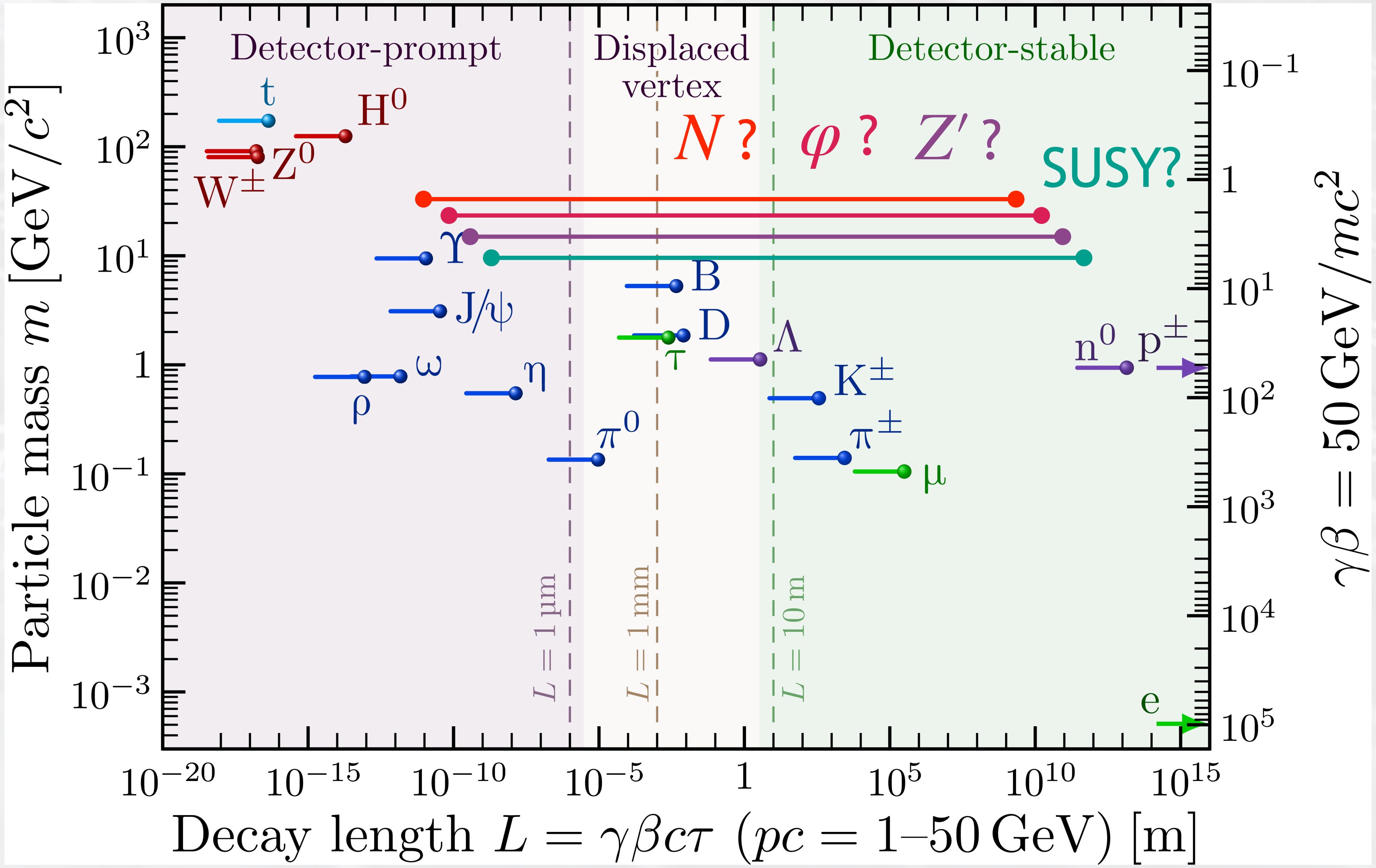
Long-lived Particles

(Beyond the Standard Model)

SM particle **lifetimes** span >30 orders of magnitude

Experiments measure "**detector-stable**" particles

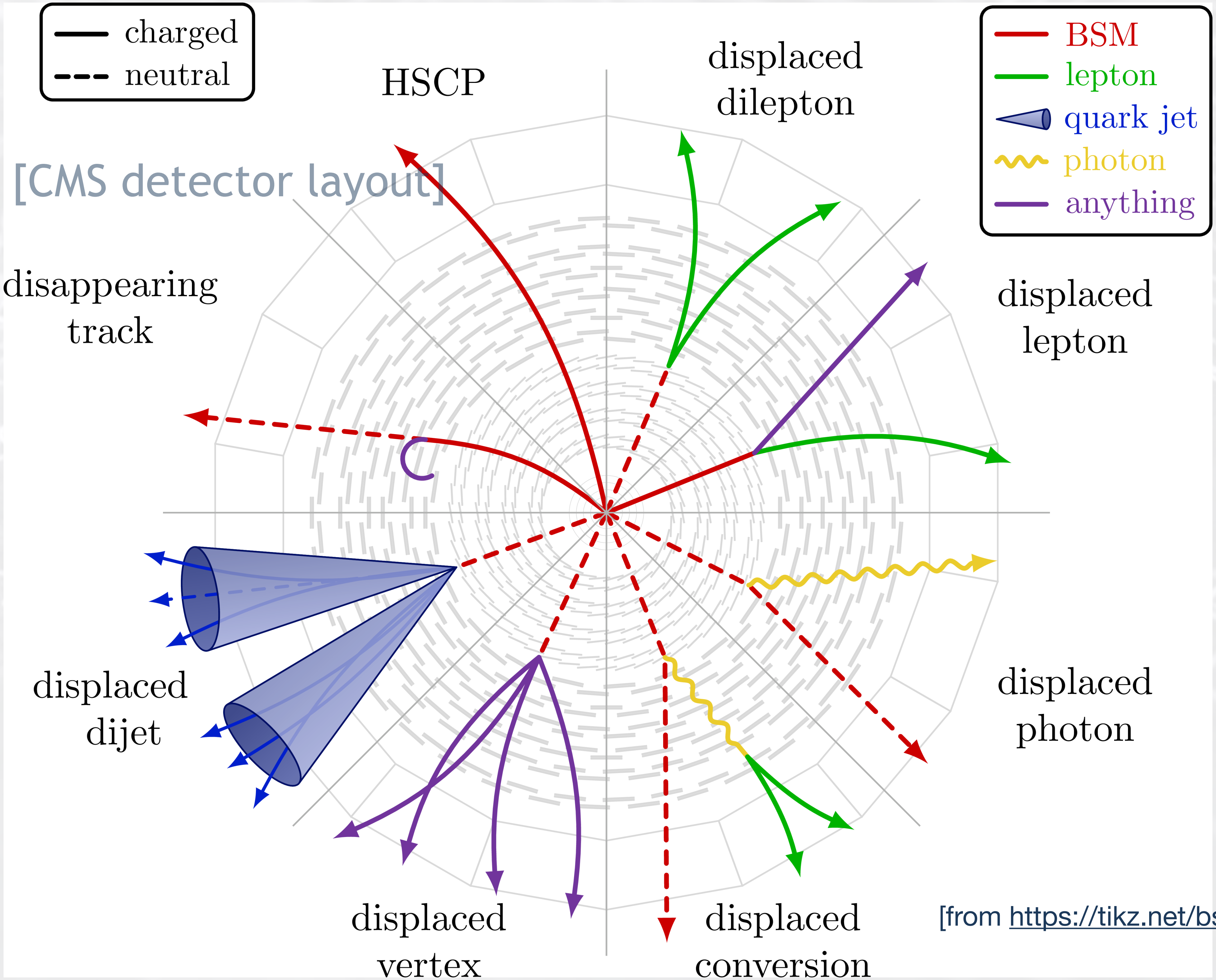
Reconstruct unstable particles from detector-stable **decay products**



LLPs appear in many **BSM** theories, dedicated LLP searches emerge as powerful probe

Cut on lifetime and reconstructed mass ⇒ low to no SM backgrounds

LLP Signatures



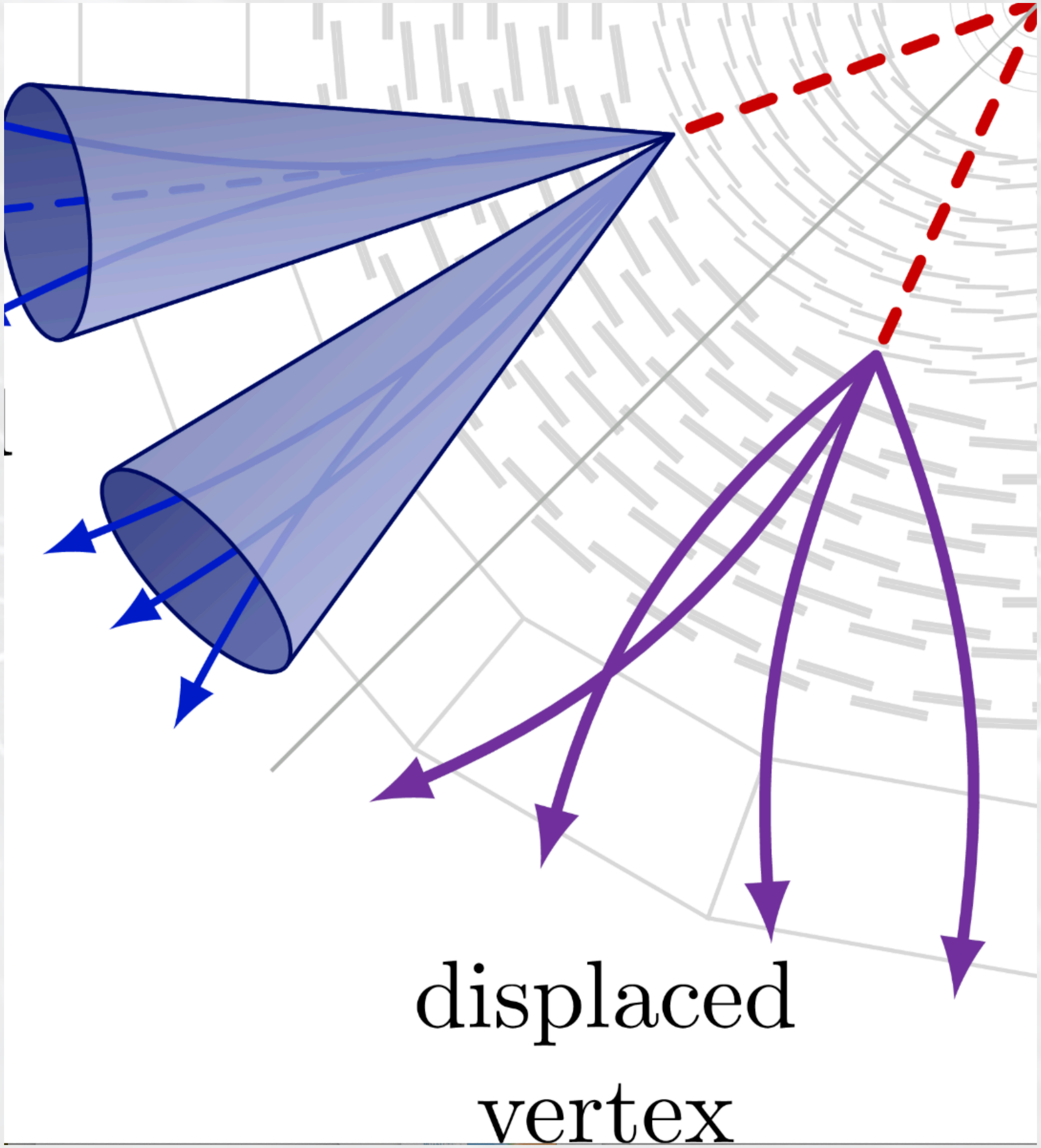
[CMS detector layout]

[from https://tikz.net/bsm_longlived/]

LLP Signatures

Focus here on **Displaced Vertex** searches

(with or without jets)



[from https://tikz.net/bsm_longlived/]

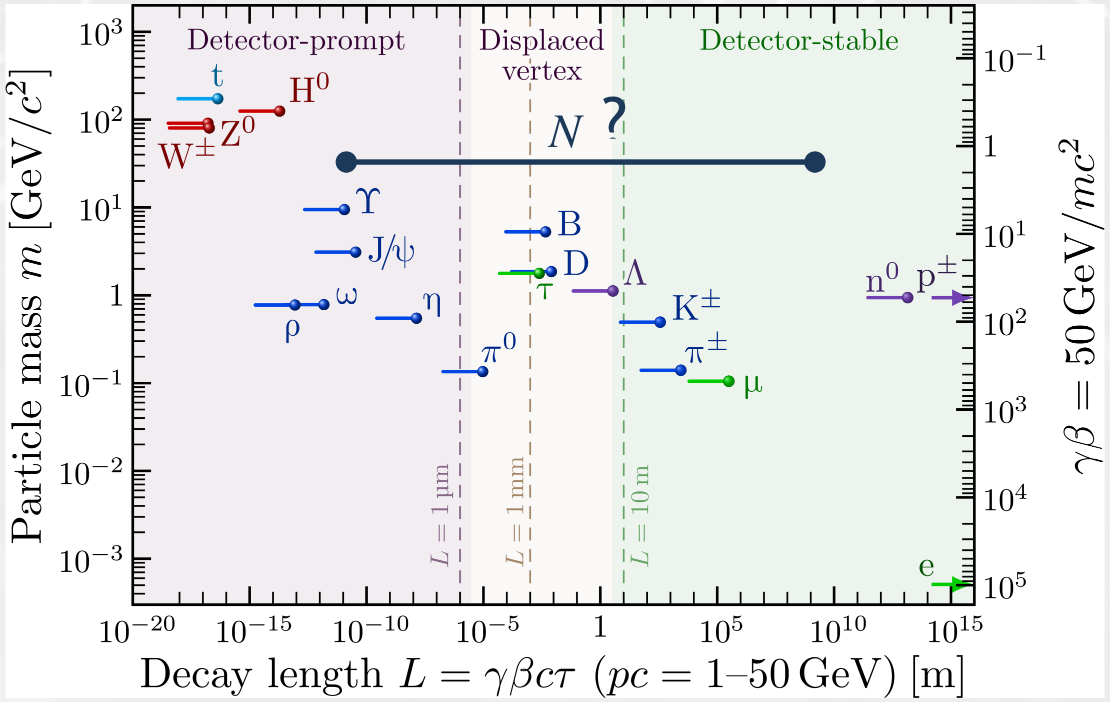
LLP Signatures

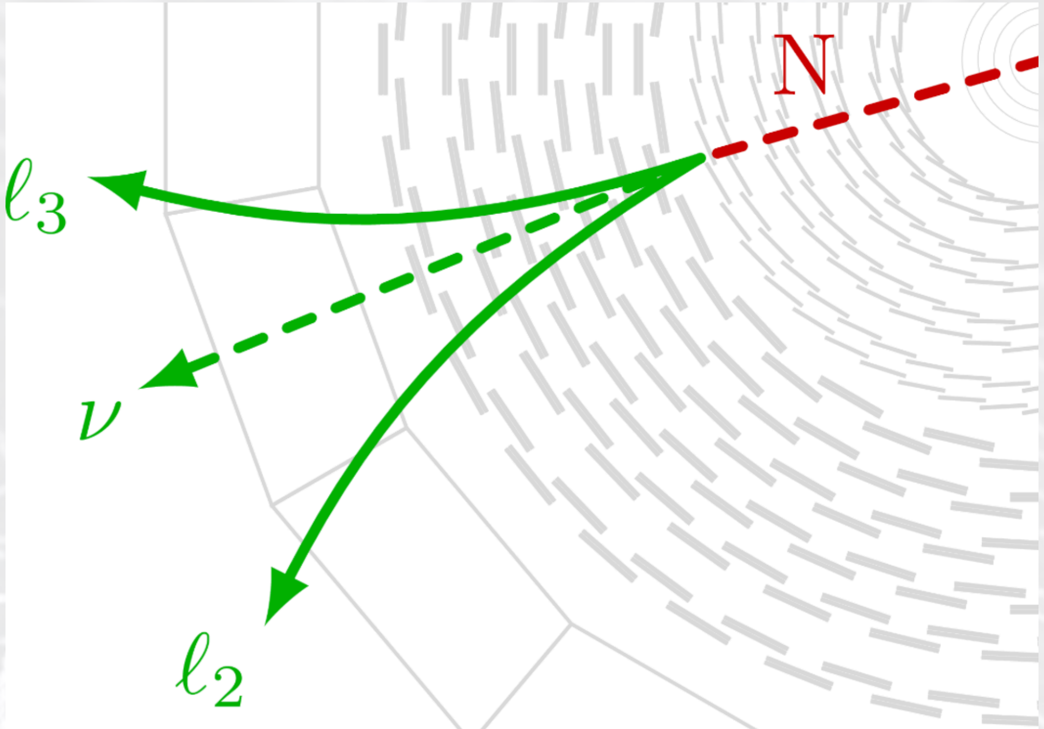
Focus here on **Displaced Vertex** searches

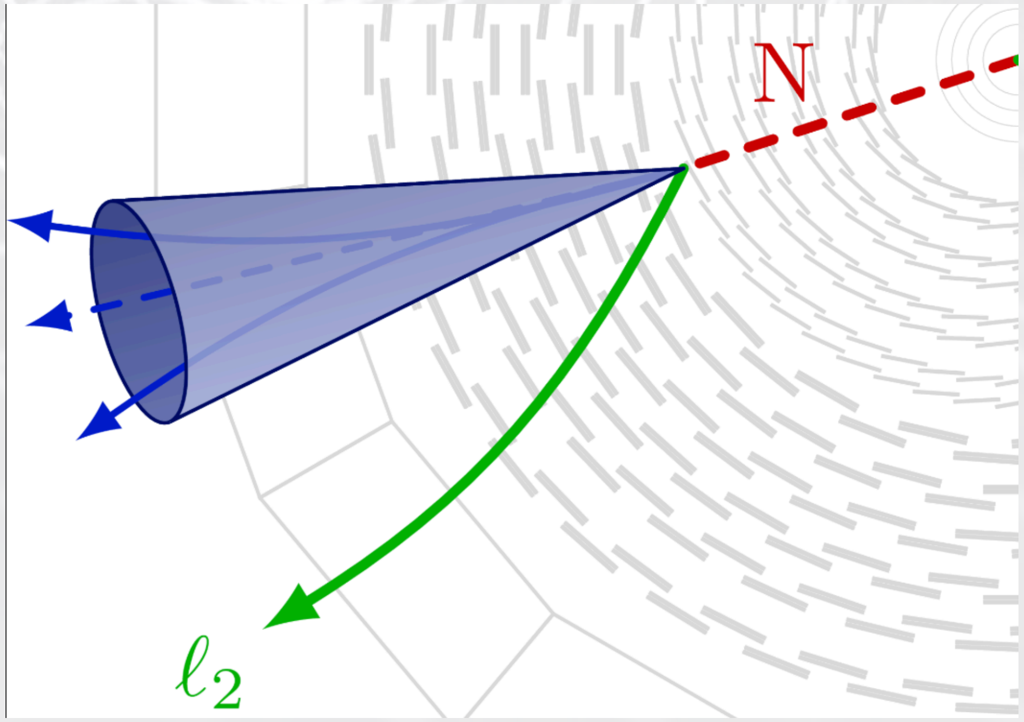
(with or without jets)

e.g. heavy neutral leptons (HNL) N:

Cut on lifetime and reconstructed mass $\Rightarrow$ **low to no SM backgrounds**

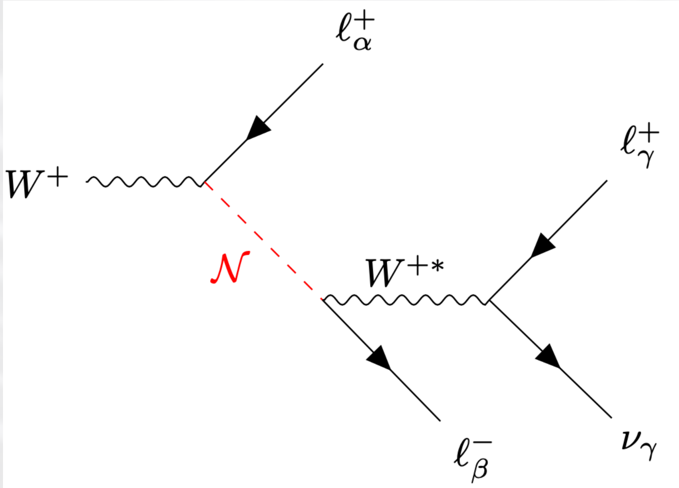






[from https://tikz.net/bsm_longlived/]

LLP searches in a nutshell:



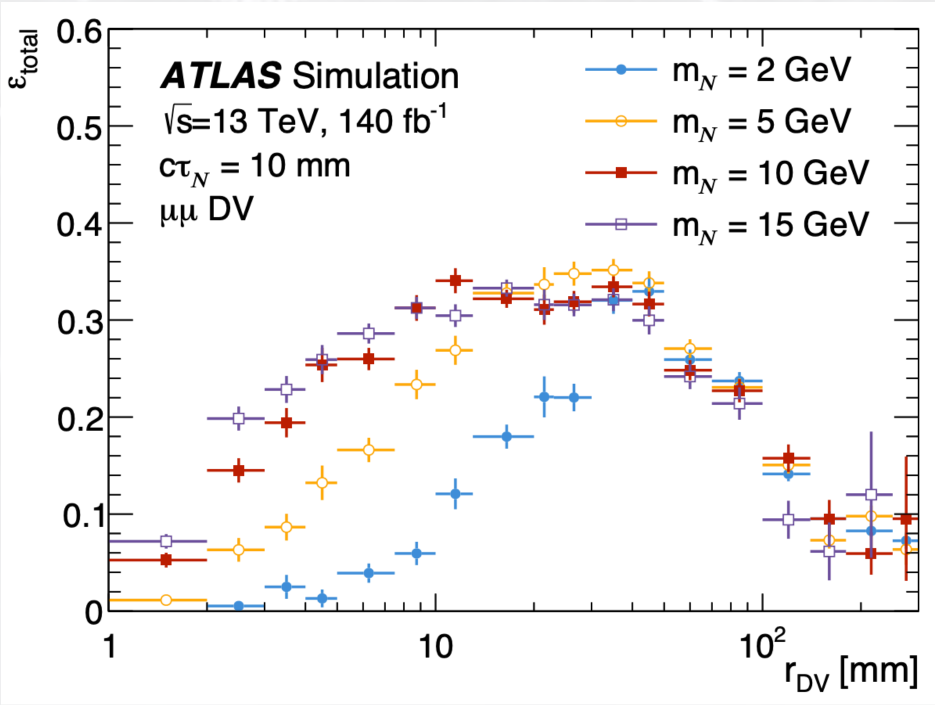
HNL N from W decay with leptonic final state

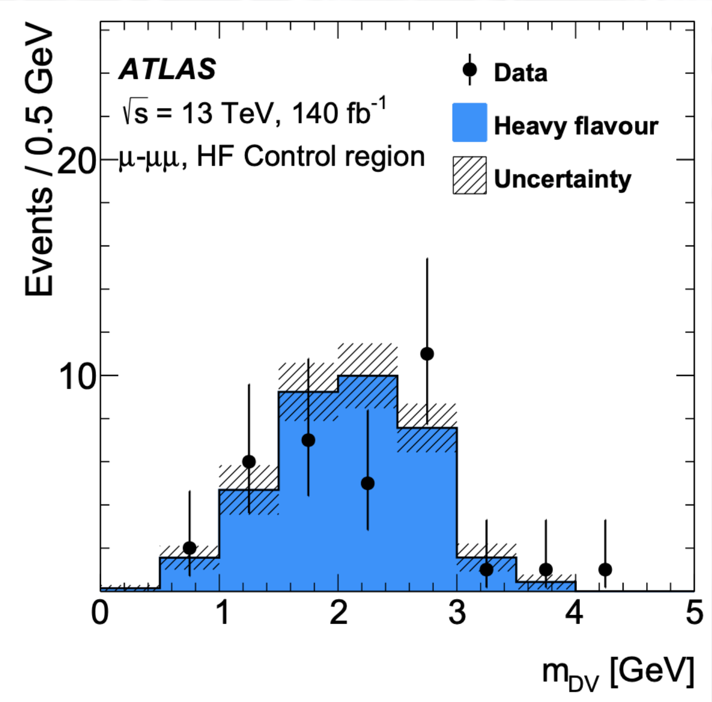
0.) Reconstruct tracks/ energy flow

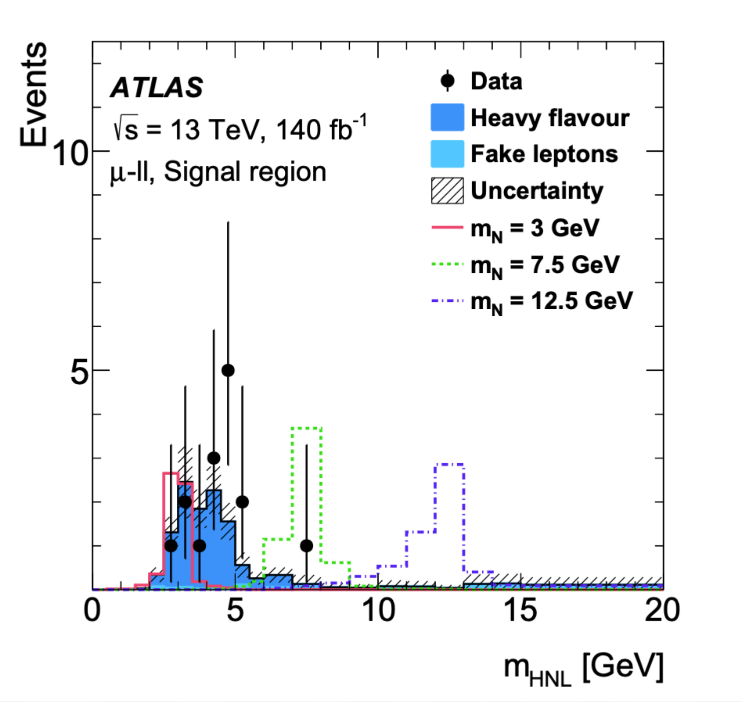
1.) Identify and fit decay **vertex**

2.) Reconstruct **vertex kinematics**

3.) Reconstruct **LLP kinematics** (constrained fit)

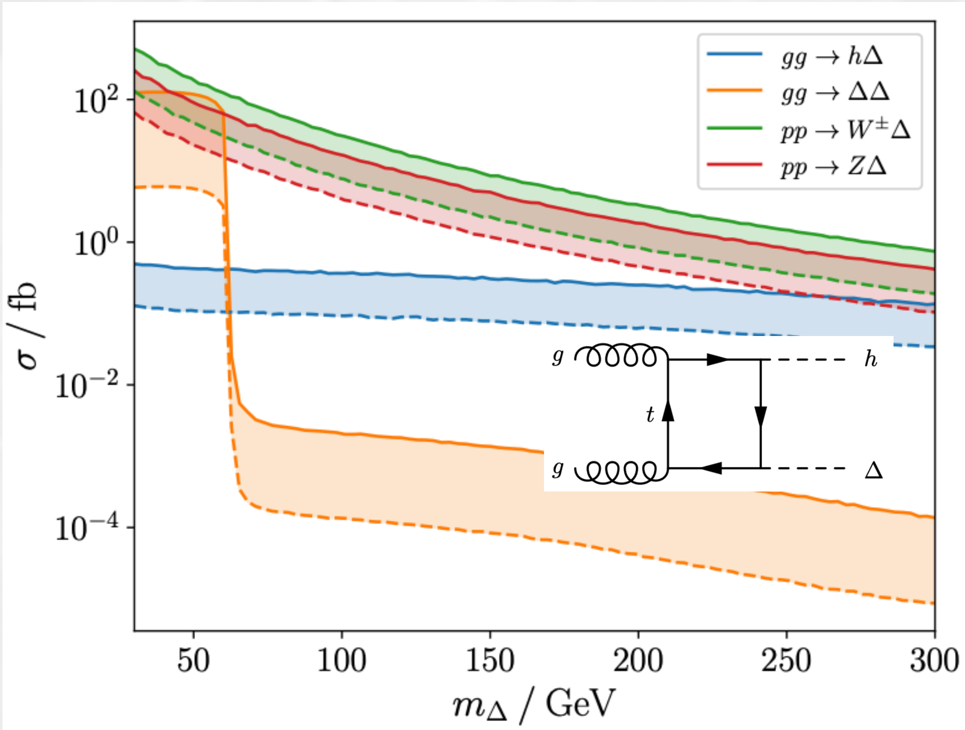




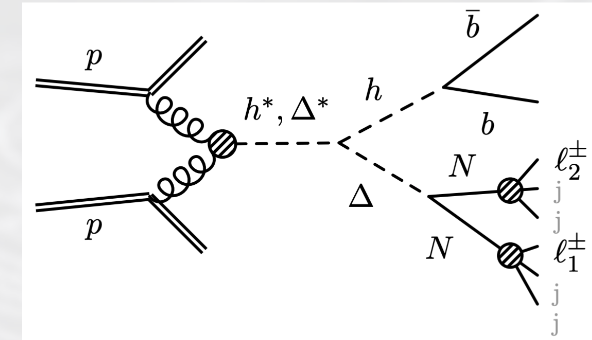


ATLAS [arXiv:2503.16213]

LLP **pheno** in a nutshell:



[from Fuks, **JK,** Nemevšek, Nesti arXiv:2503.21354]

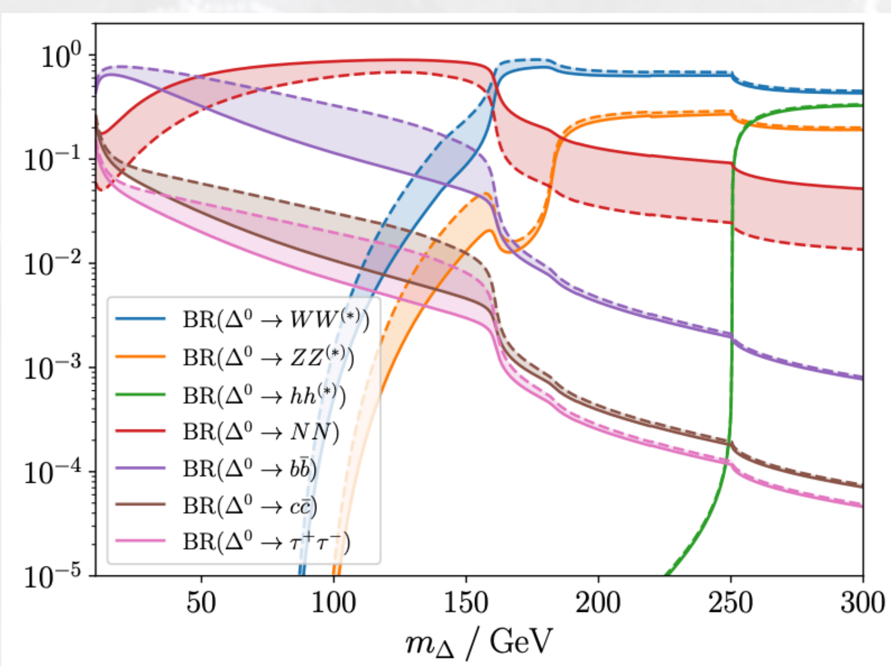


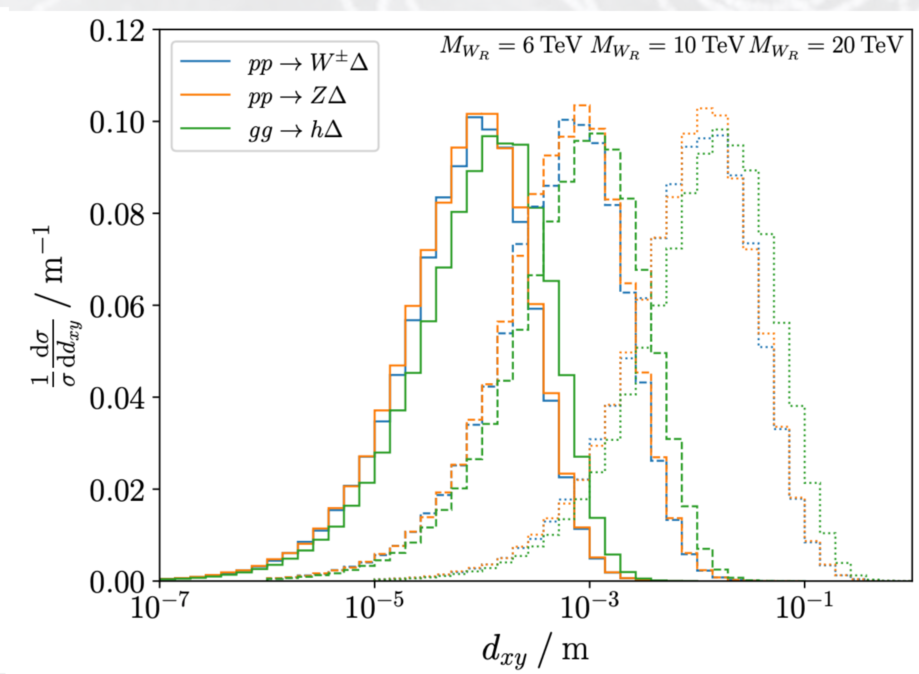
$$\mathcal{N} = \int_{d_{xy}^{\min}}^{d_{xy}^{\max}} \frac{\mathrm{d}\mathcal{N}}{\mathrm{d}(d_{xy})} = \int \mathrm{d}\Phi_3 \frac{\mathrm{d}\sigma}{\mathrm{d}\Phi_3} \left(e^{-\frac{d_{xy}^{\min}}{\langle d_{xy} \rangle}} - e^{-\frac{d_{xy}^{\max}}{\langle d_{xy} \rangle}} \right)$$

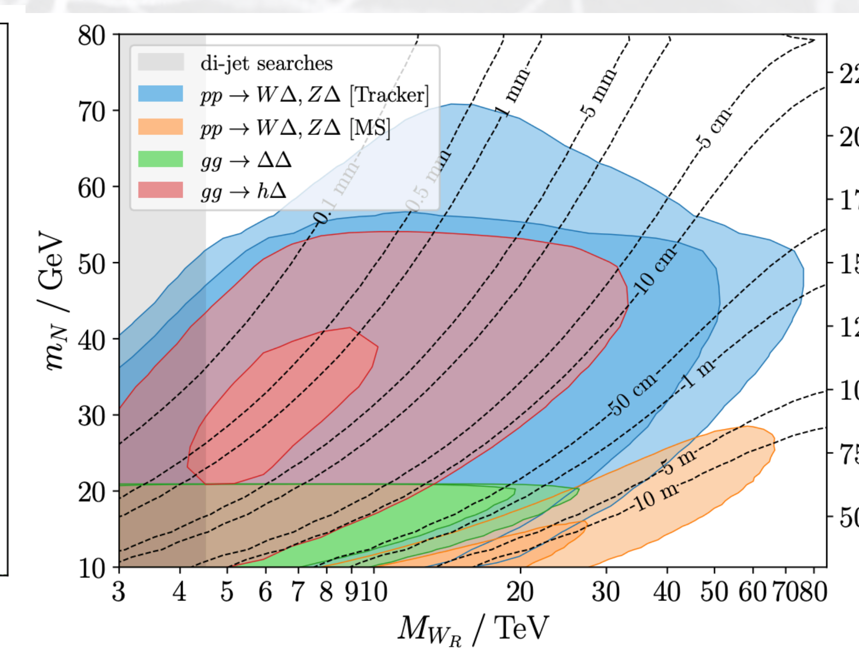
1.) Calculate $\sigma \times \mathrm{BR}$

2.) Calculate **displacement in the Lab-frame**

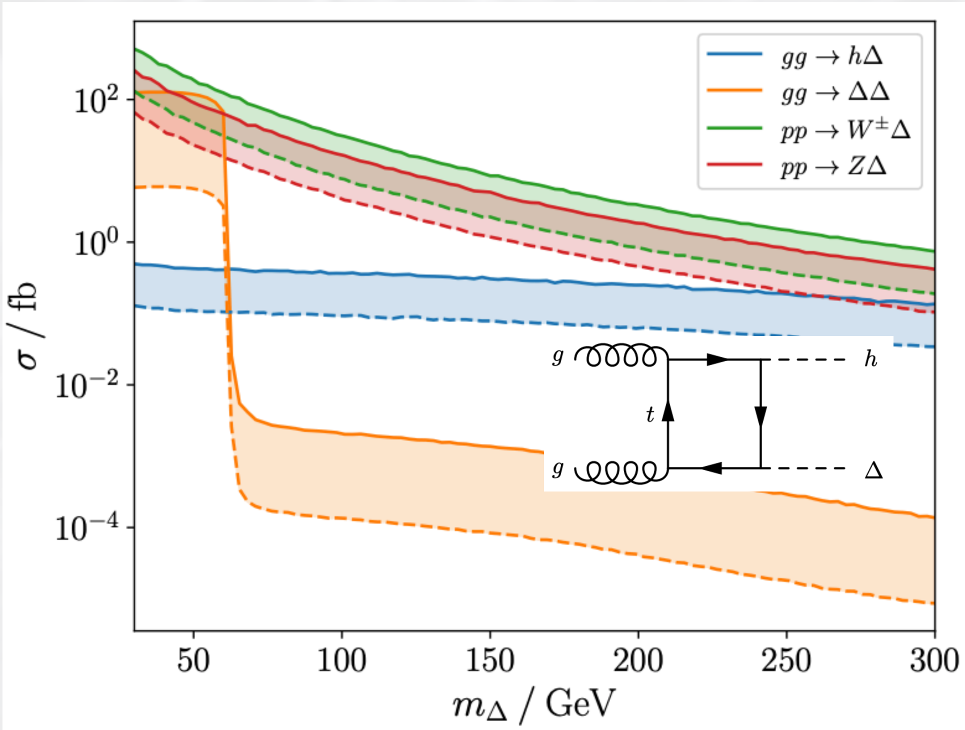
3.) **Simulate** Events & **analyse**



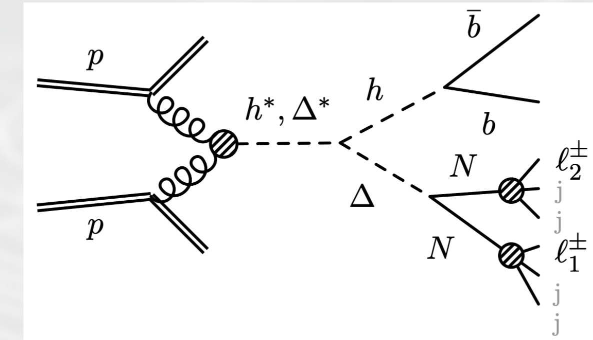




LLP pheno in a nutshell:



[from Fuks, **JK,** Nemevšek, Nesti arXiv:2503.21354]

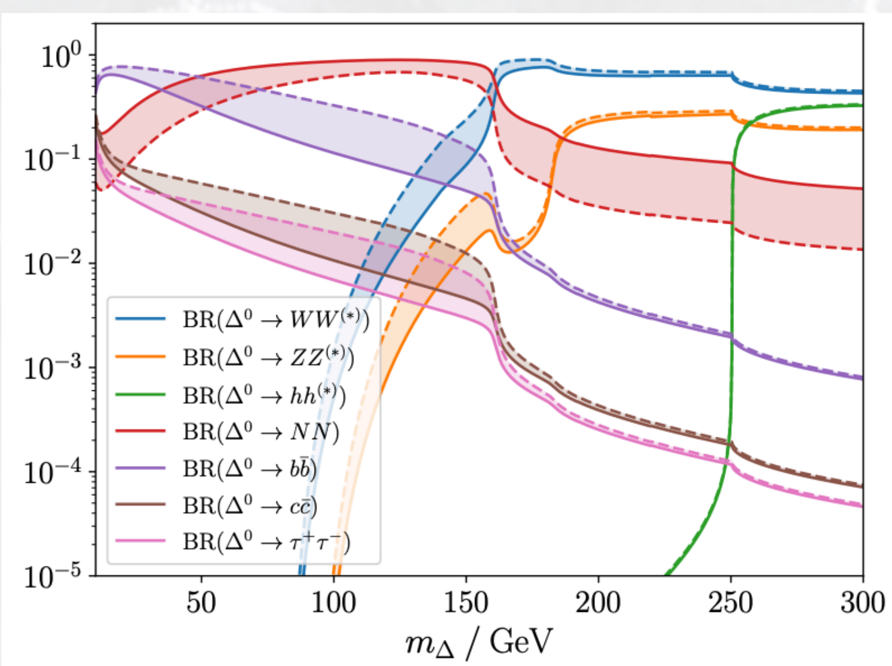


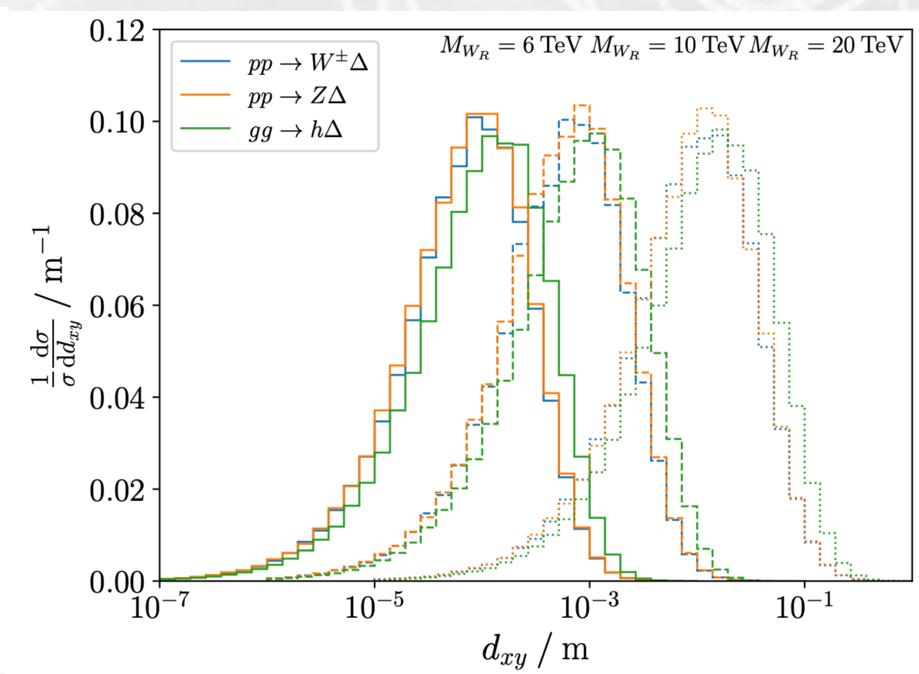
$$\mathcal{N} = \int_{d_{xy}^{\min}}^{d_{xy}^{\max}} \frac{\mathrm{d}\mathcal{N}}{\mathrm{d}(d_{xy})} = \int \mathrm{d}\Phi_3 \frac{\mathrm{d}\sigma}{\mathrm{d}\Phi_3} \left(e^{-\frac{d_{xy}^{\min}}{\langle d_{xy} \rangle}} - e^{-\frac{d_{xy}^{\max}}{\langle d_{xy} \rangle}} \right)$$

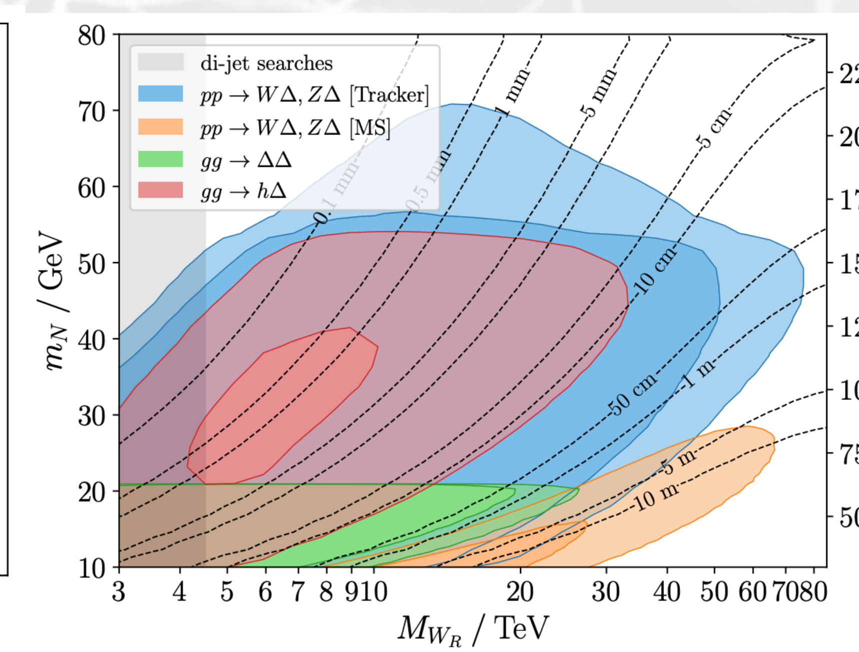
1.) Calculate $\sigma \times \mathrm{BR}$

2.) Calculate **displacement in the Lab-frame**

3.) **Assume** 100/80/X % **vertexing efficiency (experiments will take care of it right???)**







Simulation/reconstruction

Event generation: MadGraph → Pythia → Detector Sim. → analysis code

Full Geant4 detector sim. too slow/unnecessary → fast sim. with e.g. Delphes

DELPHES
fast simulation

But no **vertexing** in Delphes!

Simulation/reconstruction

Event generation: MadGraph → Pythia → **Detector Sim.** → **analysis code**

Full Geant4 detector sim. too slow/unnecessary → fast sim. with e.g. **Delphes**

DELPHES fast simulation

But no **vertexing** in **Delphes!**

Public **vertexing** software stacks:

CMS software/RAVE:

FCCAnalyses:

ACTS/ATLAS:

- Too experiment **specific**
- Too complicated to deploy for **pheno pipeline**
- Vertexing optimised for **primary vertex/close secondaries** (flavour tagging)

Simulation/reconstruction

Event generation: MadGraph → Pythia → **Detector Sim.** → **analysis code**

Full Geant4 detector sim. too slow/unnecessary → fast sim. with e.g. **Delphes**

DELPHES fast simulation

But no **vertexing** in **Delphes!**

Public **vertexin**

We (hep-ph) can and should do better!

CMS software/F

S/ATLAS:

- Too experiment **specific**
- Too complicated to deploy for **pheno pipeline**
- Vertexing optimised for **primary vertex/close secondaries** (flavour tagging)

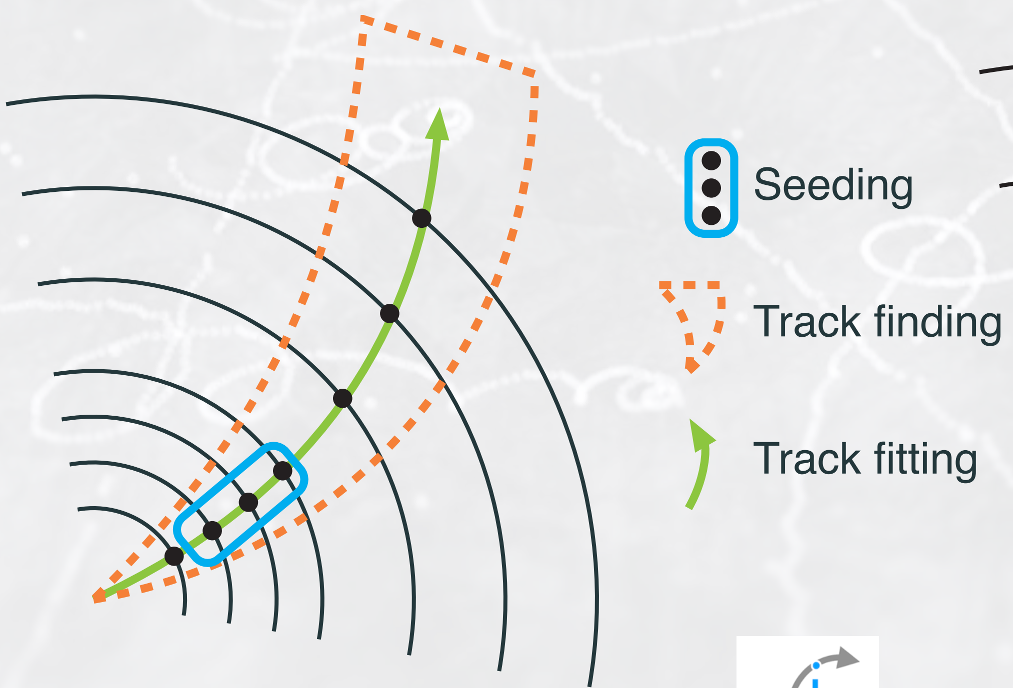
Tracking in a nutshell

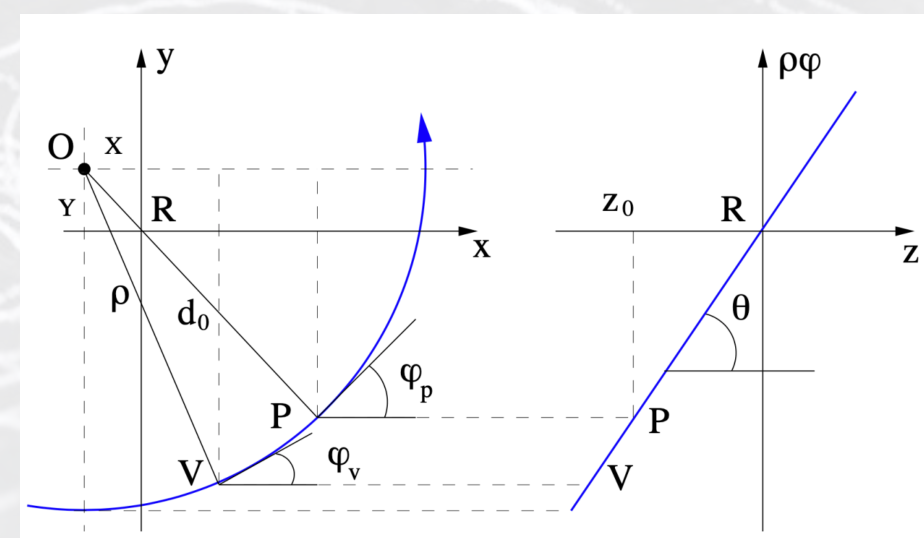
Charged particle trajectory in a (homogeneous) magnetic field ≈ helix curve

Fit track parameters to hit positions in tracker

(e.g. combinatorial Kalman Filter)

$$\vec{x}(\varphi) = \begin{pmatrix} -D_0 \sin\varphi_0 + \frac{1}{2C}(\sin(\varphi+\varphi_0) - \sin\varphi_0) \\ D_0 \cos\varphi_0 - \frac{1}{2C}(\cos(\varphi+\varphi_0) - \cos\varphi_0) \\ z_0 + \frac{\varphi\cot\theta}{2C} \end{pmatrix}$$





See e.g. Frühwirth & Strandlie 2010 & 2020 for reviews

(hardest part is pattern recognition, runs in real-time i.e. at 40 MHz)

Vertex fitting

$$\vec{x}(\varphi) = \begin{pmatrix} -D_0 \sin\varphi_0 + \frac{1}{2C}(\sin(\varphi+\varphi_0) - \sin\varphi_0) \\ D_0 \cos\varphi_0 - \frac{1}{2C}(\cos(\varphi+\varphi_0) - \cos\varphi_0) \\ z_0 + \frac{\varphi \cot\theta}{2C} \end{pmatrix}$$

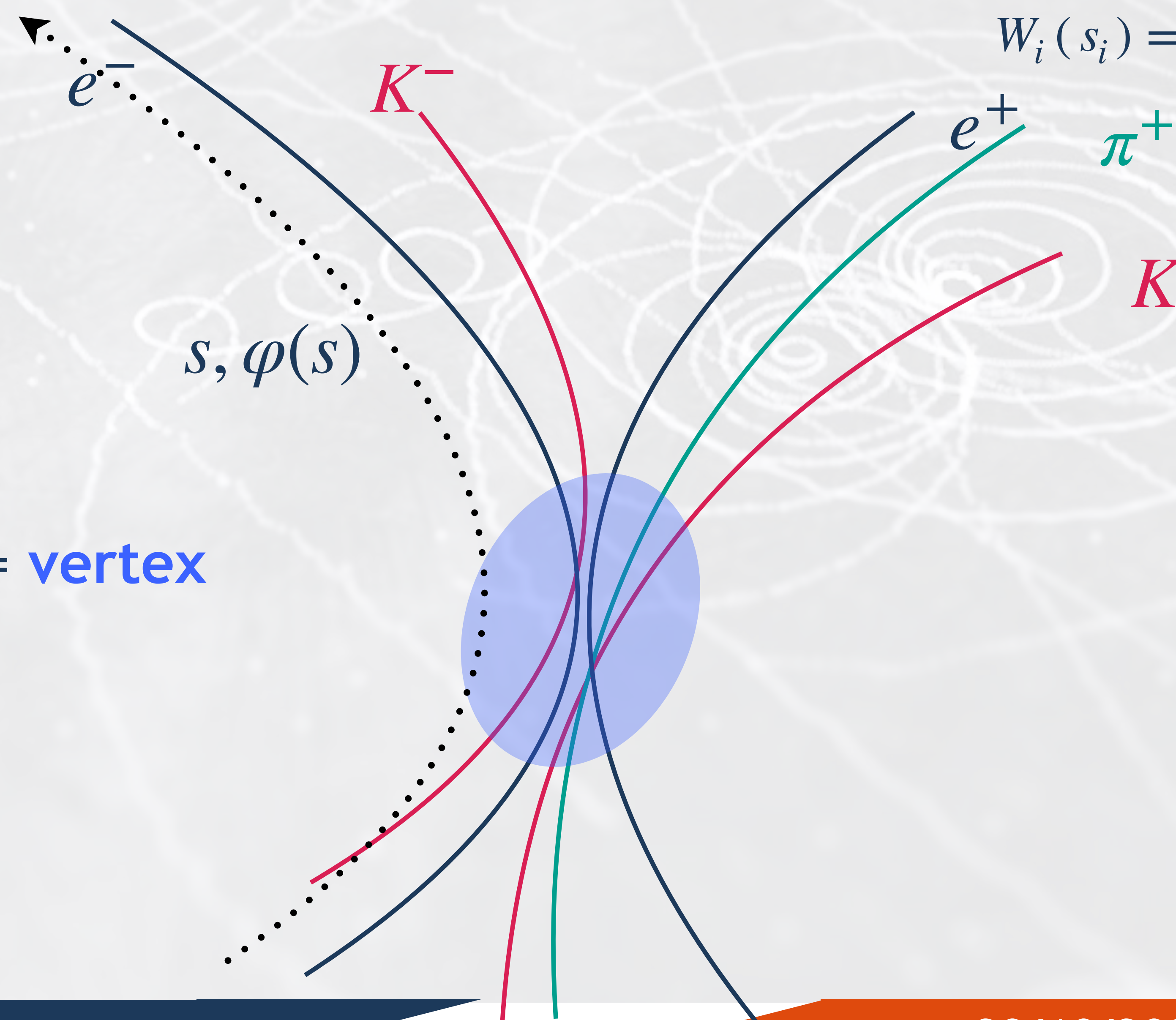
$$\varphi(s) = \frac{2C}{\sqrt{1+\cot^2\theta}} s$$

Fit tracks to a common vertex, minimise:

$$F(\{s_i\}, \vec{v}) = \frac{1}{2}\left[\sum_i^{N_{\text{tracks}}} (\vec{v} - \vec{x}_i(s_i))^T W_i(s) (\vec{v} - \vec{x}_i(s_i))\right]$$

(+ prior, e.g. beamspot)

$$W_i(s_i) = [J_{x\alpha}(s_i)\, C_\alpha\, J^T_{x\alpha}(s_i)]^{-1}$$



Find common origin = vertex

Vertex fitting

$$\vec{x}(\varphi) = \begin{pmatrix} -D_0 \sin\varphi_0 + \frac{1}{2C}(\sin(\varphi+\varphi_0) - \sin\varphi_0) \\ D_0 \cos\varphi_0 - \frac{1}{2C}(\cos(\varphi+\varphi_0) - \cos\varphi_0) \\ z_0 + \frac{\varphi\cot\theta}{2C} \end{pmatrix} \qquad \varphi(s) = \frac{2C}{\sqrt{1+\cot^2\theta}} s$$

Fit tracks to a common vertex, minimise: $F(\{s_i\}, \vec{v}) = \frac{1}{2}\left[\sum_i^{N_{\text{tracks}}} (\vec{v} - \vec{x}_i(s_i))^T W_i(s) (\vec{v} - \vec{x}_i(s_i))\right]$ (+ prior, e.g. beamspot)

$$W_i(s_i) = [\, J_{x\alpha}(s_i)\, C_\alpha\, J^T_{x\alpha}(s_i)\,]^{-1}$$

Gauss-Newton algorithm: solve $\mathscr{H}\Delta\vec{\alpha} = -g$, $\mathscr{H}$ = Hessian, g = gradient

Inverting the Hessian scales as $\mathcal{O}((N+3)^3)$:(

$$\begin{pmatrix} H_{vv} & H_{vs} \\ H_{sv} & H_{ss} \end{pmatrix} \begin{pmatrix} \Delta\vec{v} \\ \Delta s_i \end{pmatrix} = -\begin{pmatrix} g_v \\ g_s \end{pmatrix}$$

$$g_v = \left[\sum_i W_i^0 \vec{r}_i^{\,0}\right] + \tilde{W}(\vec{v}_0 - \vec{v}_p), \quad g_s^i = -\vec{t}_i^{\,0,T} W_i^0 \vec{r}_i^{\,0}$$

$$H_{vv} = \left[\sum_i W_i^0\right] + \tilde{W}, \quad H_{ss} = \mathrm{diag}(\vec{t}_1^{\,0,T} W_1^0 \vec{t}_1^{\,0}, \vec{t}_2^{\,0,T} W_2^0 \vec{t}_2^{\,0}, \ldots, \vec{t}_N^{\,0,T} W_N^0 \vec{t}_N^{\,0})$$

$$H_{vs} = H_{sv}^T = -(W_1^0 \vec{t}_1^{\,0}, W_2^0 \vec{t}_2^{\,0}, \ldots, W_N^0 \vec{t}_N^{\,0}) .$$

Vertex fitting

$$\vec{x}(\varphi) = \begin{pmatrix} -D_0 \sin\varphi_0 + \frac{1}{2C}(\sin(\varphi+\varphi_0) - \sin\varphi_0) \\ D_0\cos\varphi_0 - \frac{1}{2C}(\cos(\varphi+\varphi_0) - \cos\varphi_0) \\ z_0 + \frac{\varphi\cot\theta}{2C} \end{pmatrix} \qquad \varphi(s) = \frac{2C}{\sqrt{1+\cot^2\theta}} s$$

Fit tracks to a common vertex, minimise: $F(\{s_i\}, \vec{v}) = \frac{1}{2}\left[\sum_i^{N_{\text{tracks}}} (\vec{v} - \vec{x}_i(s_i))^T W_i(s)\,(\vec{v} - \vec{x}_i(s_i)) \right]$ (+ prior, e.g. beamspot)

Gauss-Newton algorithm: solve $\mathscr{H}\Delta\overrightarrow{\alpha} = -g$, $\mathscr{H}$ = Hessian, g = gradient

Inverting the Hessian scales as $\mathcal{O}((N+3)^3)$:(

Split system into vertex & phases:

$$\begin{pmatrix} H_{vv} & H_{vs} \\ H_{sv} & H_{ss} \end{pmatrix} \begin{pmatrix} \Delta\vec{v} \\ \Delta s_i \end{pmatrix} = -\begin{pmatrix} g_v \\ g_s \end{pmatrix}$$

$\Delta\vec{v} = \vec{v} - \vec{v}_0,$

$\vec{v}_0 \sim$ initial guess

$H_{ss}^{N\times N} \propto \text{diag}, \quad H_{sv}^{N\times 3}, \quad H_{vv}^{3\times 3}$

1.) Optimise phases: scales as $\mathcal{O}(N)$:)

$$\Delta s_i = -H_{ss}^{-1}(g_s^i + H_{sv}\,\Delta v)$$

2.) Then vertex: scales as $\mathcal{O}(3^3)$

$$\vec{v}^\star = \left(\tilde{W} + \sum_i W_i^\perp\right)^{-1}\left(\tilde{W}\vec{v}_p + \sum_i W_i^\perp \vec{x}_i(s_i^0)\right)$$

[see Billoir, Frühwirth, Regler 1985, Billoir & Qian 1992]

Vertex fitting

$$\vec{x}(\varphi) = \begin{pmatrix} -D_0 \sin\varphi_0 + \frac{1}{2C}(\sin(\varphi+\varphi_0) - \sin\varphi_0) \\ D_0 \cos\varphi_0 - \frac{1}{2C}(\cos(\varphi+\varphi_0) - \cos\varphi_0) \\ z_0 + \frac{\varphi \cot\theta}{2C} \end{pmatrix} \qquad \varphi(s) = \frac{2C}{\sqrt{1+\cot^2\theta}} s$$

Fit tracks to a common vertex, minimise: $F(\{s_i\}, \vec{v}) = \frac{1}{2}\left[\sum_i^{N_{\text{tracks}}} (\vec{v} - \vec{x}_i(s_i))^T W_i(s) (\vec{v} - \vec{x}_i(s_i))\right]$

scales as $\mathcal{O}(N^3)$:(

Gauss-Newton algorithm: solve $\mathcal{H}\Delta\vec{\alpha} = -g$, $\mathcal{H}$ = Hessian, g = gradient

Split system

$\Delta\vec{v} = \vec{v} - \vec{v}_0$,

$\vec{v}_0 \sim$ initial guess

$$\dots \begin{matrix} g_v \\ g_s \end{matrix}\Bigg)$$

$H_{ss}^{N\times N} \propto \text{diag}, \quad H_{sv}^{N\times 3}, \quad H_{vv}^{3\times 3}$

Vertex fitting has been conclusively solved in the 80's...

1.) Optimise phases: scales as $\mathcal{O}(N)$:)

$$\Delta s_i = -H_{ss}^{-1}(g_s^i + H_{sv}\,\Delta v)$$

2.) Then vertex: scales as $\mathcal{O}(3^3)$

$$\vec{v}^\star = \left(\tilde{W} + \sum_i W_i^\perp\right)^{-1} \left(\tilde{W}\vec{v}_p + \sum_i W_i^\perp \vec{x}_i(s_i^0)\right)$$

[see Billoir, Frühwirth, Regler 1985, Billoir & Qian 1992]

Vertex fitting

$$\vec{x}(\varphi) = \begin{pmatrix} -D_0 \sin\varphi_0 + \frac{1}{2C}(\sin(\varphi+\varphi_0) - \sin\varphi_0) \\ D_0 \cos\varphi_0 - \frac{1}{2C}(\cos(\varphi+\varphi_0) - \cos\varphi_0) \\ z_0 + \frac{\varphi\cot\theta}{2C} \end{pmatrix} \qquad \varphi(s) = \frac{2C}{\sqrt{1+\cot^2\theta}}s$$

Fit tracks to a common vertex, minimise: $$F(\{s_i\},\vec{v}) = \frac{1}{2}\left[\sum_i^{N_{\text{tracks}}} (\vec{v} - \vec{x}_i(s_i))^T W_i(s)\,(\vec{v} - \vec{x}_i(s_i))\right]$$

scales as $\mathcal{O}(N^3)$:(

Gauss-Newton algorithm: solve $\mathcal{H}\,\Delta\overrightarrow{\alpha} = -g$, $\mathcal{H}$ = Hessian, g = gradient

Are you SCHUR?

Split system i

$\Delta\vec{v} = \vec{v} - \vec{v}_0$,

$\vec{v}_0 \sim$ initial guess

$H_{ss}^{N\times N} \propto \text{diag}, \quad H_{sv}^{N\times 3}, \quad H_{vv}^{3\times 3}$

1.) Optimise phases: scales as $\mathcal{O}(N)$:)

$$\Delta s_i = -H_{ss}^{-1}(g_s^i + H_{sv}\,\Delta v)$$

2.) Then vertex: scales as $\mathcal{O}(3^3)$

$$\vec{v}^\star = \left(\tilde{W} + \sum_i W_i^\perp\right)^{-1}\left(\tilde{W}\vec{v}_p + \sum_i W_i^\perp \vec{x}_i(s_i^0)\right)$$

[see Billoir, Frühwirth, Regler 1985, Billoir & Qian 1992]

Initialisation-free vertex fitting

JK [arXiv:2510.00856]

$$\begin{pmatrix} H_{vv} & H_{vs} \\ H_{sv} & H_{ss} \end{pmatrix} \begin{pmatrix} \Delta\vec{v} \\ \Delta s_i \end{pmatrix} = - \begin{pmatrix} g_v \\ g_s \end{pmatrix}$$

Splitting the **Hessian** $\simeq$ **Schur complement**

Alternative:

1.) Eliminate vertex:

$$\Delta\vec{v} = -H_{vv}^{-1}(g_v + H_{vs}\Delta s)$$

2.) Solve for phases: scales as $\mathcal{O}(N^3)$:(

$$(H_{ss} - H_{sv}H_{vv}^{-1}H_{vs})^{N\times N}\Delta s = -(g_s - H_{sv}H_{vv}^{-1}g_v)$$

Initialisation-free vertex fitting

JK [arXiv:2510.00856]

Splitting the **Hessian** $\simeq$ **Schur complement**

$$\begin{pmatrix} H_{vv} & H_{vs} \\ H_{sv} & H_{ss} \end{pmatrix} \begin{pmatrix} \Delta\vec{v} \\ \Delta s_i \end{pmatrix} = - \begin{pmatrix} g_v \\ g_s \end{pmatrix}$$

Alternative:

1.) Eliminate vertex:

$$\Delta\vec{v} = -H_{vv}^{-1}(g_v + H_{vs}\Delta s)$$

2.) Solve for phases: scales as $\mathcal{O}(N^3)$:(

$$(H_{ss} - H_{sv}H_{vv}^{-1}H_{vs})^{N\times N}\Delta s = -(g_s - H_{sv}H_{vv}^{-1}g_v)$$

"Diagonal + Rank-3 update"

$\Rightarrow$ Use **Woodbury identity**:

scales also as $\mathcal{O}(N)$!!!

$$(H_{ss} - H_{sv}H_{vv}^{-1}H_{vs})^{-1} = H_{ss}^{-1} + H_{ss}^{-1}H_{sv}(H_{vv} - H_{vs}H_{ss}^{-1}H_{sv})^{-1}H_{vs}H_{ss}^{-1}$$

Traditional method: 1.) **profile** the **phases** (nuisance parameters), 2.) get the **vertex**

New method: 1.) **profile** the vertex, 2.) optimise the **phases**

Schur complements of the same Hessian $\Rightarrow$ **stat. & algebraic equivalence** (of **MLE**)

Initialisation-free vertex fitting

JK [arXiv:2510.00856]

$$\begin{pmatrix} H_{vv} & H_{vs} \\ H_{sv} & H_{ss} \end{pmatrix} \begin{pmatrix} \Delta \vec{v} \\ \Delta s_i \end{pmatrix} = - \begin{pmatrix} g_v \\ g_s \end{pmatrix}$$

Splitting the **Hessian** $\simeq$ **Schur complement**

Alternative:

1.) Eliminate vertex:

$$\Delta \vec{v} = - H_{vv}^{-1}(g_v + H_{vs}\Delta s)$$

2.) Solve for phases: scales as $\mathcal{O}(N^3)$:(

$$(H_{ss} - H_{sv}H_{vv}^{-1}H_{vs})^{N\times N}\Delta s = -(g_s - H_{sv}H_{vv}^{-1}g_v)$$

"Diagonal + Rank-3 update"

$\Rightarrow$ After eliminating the vertex: no need for **initial guess!**

$$\vec{v}^\star(\{s_i\}) = \tilde{S}^{-1}\left(\tilde{W}\vec{v}_p + \sum_i W_i \vec{x}_i\right)$$

$$F(\{s_i\}, \vec{v}^\star(\{s_i\})) = \frac{1}{4}\sum_{i<j}(\vec{x}_i - \vec{x}_j)^T \left(W_i \tilde{S}^{-1} W_j + W_j \tilde{S}^{-1} W_i\right)(\vec{x}_i - \vec{x}_j)$$

Traditional: **guess** the **vertex**, check track compatibility, move the **vertex**, repeat

New method: tracks determine the **vertex** as **mutual closest point**

Initialisation-free vertex fitting

JK [arXiv:2510.00856]

Splitting the **Hessian** $\simeq$ **Schur complement**

$$\begin{pmatrix} H_{vv} & H_{vs} \\ H_{sv} & H_{ss} \end{pmatrix} \begin{pmatrix} \Delta\vec{v} \\ \Delta s_i \end{pmatrix} = - \begin{pmatrix} g_v \\ g_s \end{pmatrix}$$

Alternative:

1.) Eliminate vertex:

$$\Delta\vec{v} = - H_{vv}^{-1}(g_v + H_{vs}\Delta s)$$

2.) Solve for phases: scales as $\mathcal{O}(N^3)$:(

$$(H_{ss} - H_{sv}H_{vv}^{-1}H_{vs})^{N\times N}\Delta s = - (g_s - H_{sv}H_{vv}^{-1}g_v)$$

"Diagonal + Rank-3 update"

$\Rightarrow$ After eliminating the vertex: no need for **initial guess!**

$$\vec{v}^\star(\{s_i\}) = \tilde{S}^{-1}\left(\tilde{W}\vec{v}_p + \sum_i W_i\vec{x}_i\right)$$

$$F(\{s_i\}, \vec{v}^\star(\{s_i\})) = \frac{1}{4}\sum_{i<j}(\vec{x}_i - \vec{x}_j)^T\left(W_i\tilde{S}^{-1}W_j + W_j\tilde{S}^{-1}W_i\right)(\vec{x}_i - \vec{x}_j)$$

Traditional: **guess** the **vertex**, check track compatibility, move the **vertex**, repeat

New method: tracks determine the **vertex** as **mutual closest point**

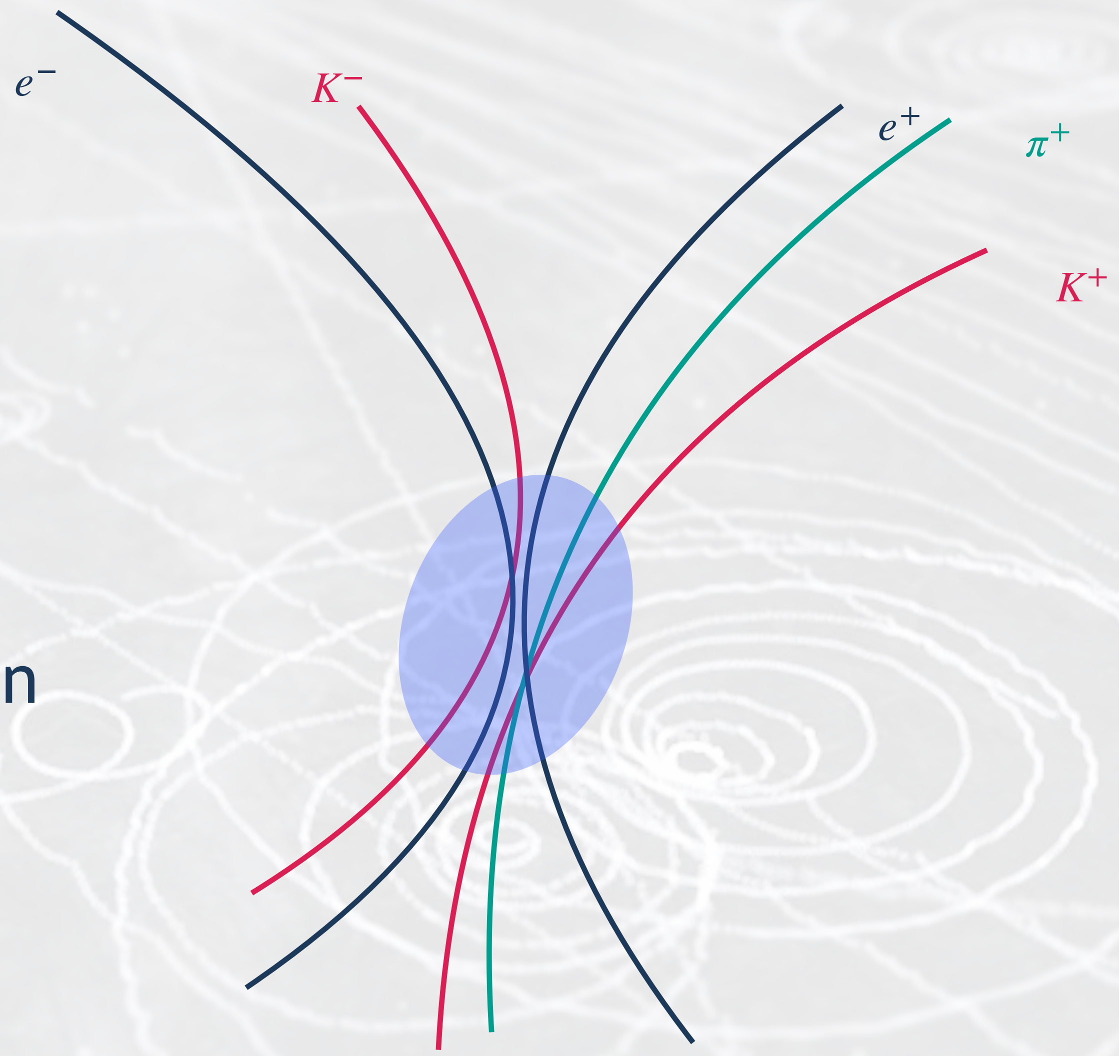
Both **converge near-quadratically,** but **new method** has practically **global basin**

Vertex finding

Find common origin = **vertex**

$\Rightarrow$ just fit the tracks

Use **weighted fit** for outlier detection



Vertex finding

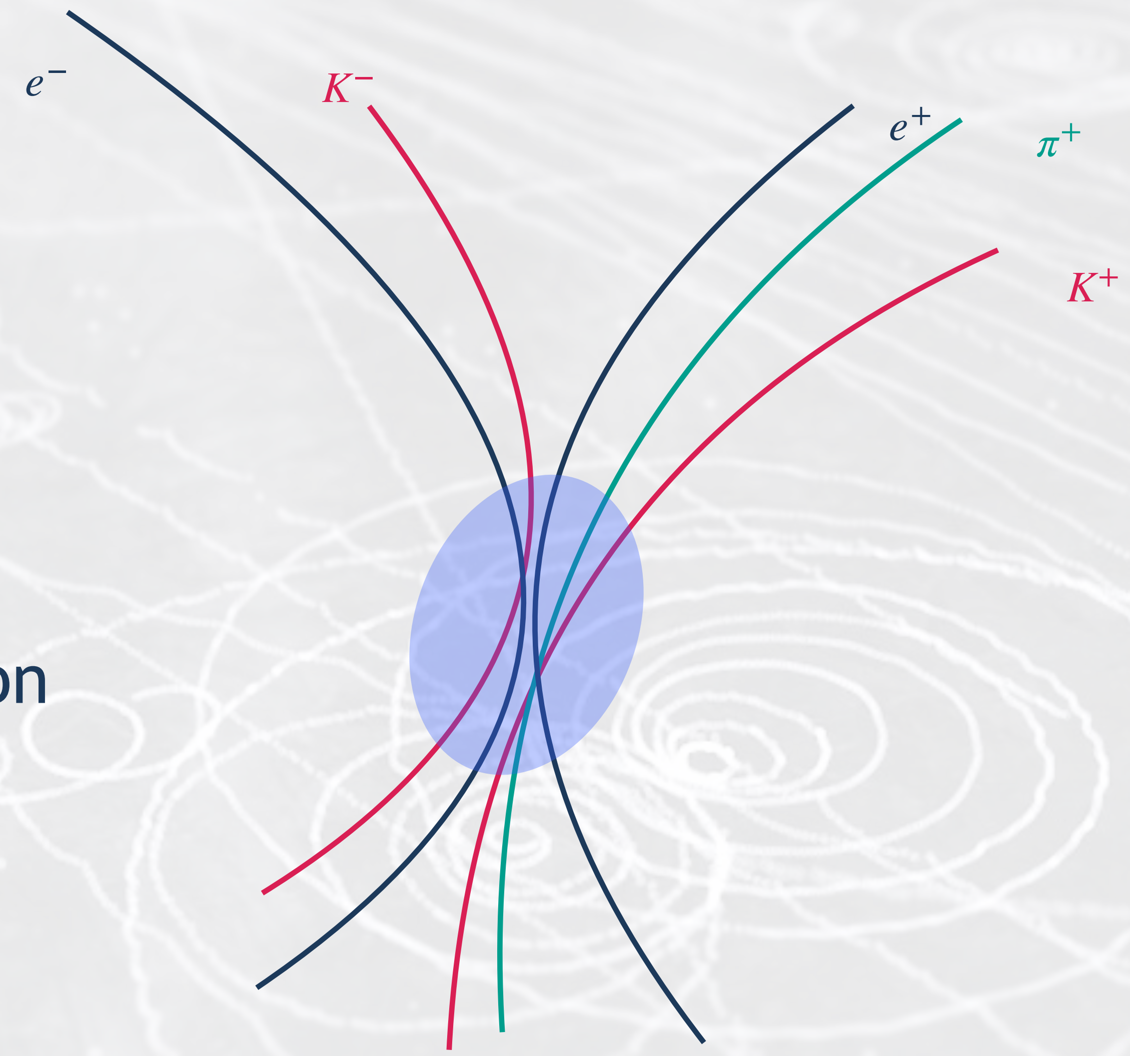
Find common origin = **vertex**

$\Rightarrow$ just fit the tracks

Use **weighted fit** for outlier detection

e.g. $B_s \to \phi(\to K^+K^-)e^+e^-$:

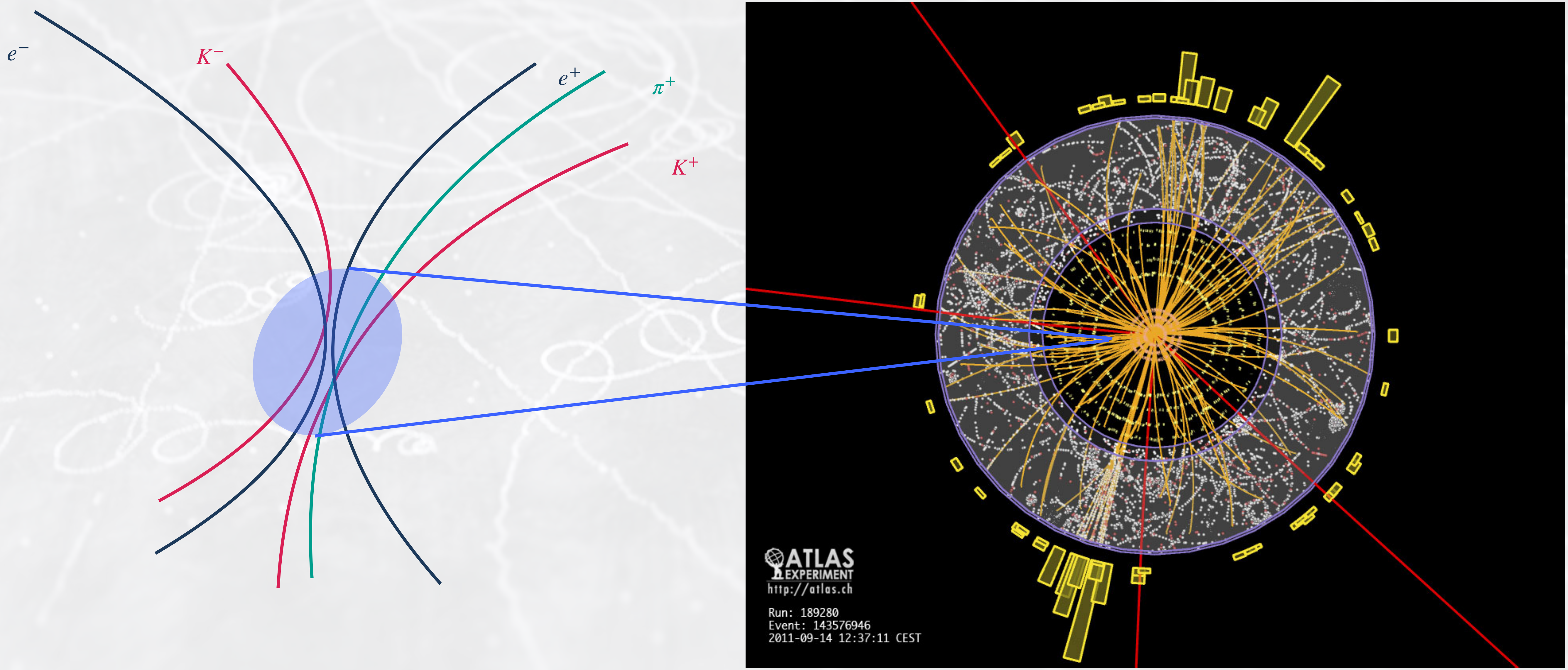
π^+ doesn't belong and get's **down-weighted/removed**



Vertex finding

But real events look like this:

$\mathcal{O}(100-1000)$ tracks / event!



Need to determine a set of tracks to be fitted $\Rightarrow$ **pattern recognition**

Vertex finding

Need to determine a set of tracks to be fitted ⇒ **pattern recognition**

Traditional (**iterative**) methods: optimised for PV finding in **high pile-up**

close secondaries for **flavour tagging**

⇒ not suitable for **exotic LLP topologies**

(recently also optimised graph and GNN methods by ATLAS & CMS)

Vertex finding

Need to determine a set of tracks to be fitted ⇒ **pattern recognition**

Traditional (**iterative**) methods: optimised for PV finding in **high pile-up**

close secondaries for **flavour tagging**

⇒ not suitable for **exotic LLP topologies**

(recently also optimised graph and GNN methods by ATLAS & CMS)

Graph-based approach: tracks are **nodes of compatibility graph**

JK [arXiv:2510.00856]

Vertex finding

Need to determine a set of tracks to be fitted ⇒ **pattern recognition**

Traditional (**iterative**) methods: optimised for PV finding in **high pile-up**
close secondaries for **flavour tagging**

⇒ not suitable for **exotic LLP topologies**

(recently also optimised graph and GNN methods by ATLAS & CMS)

Graph-based approach: tracks are **nodes of compatibility graph**

JK [arXiv:2510.00856]

Build **edges** if $\chi^2_2 \lesssim 9$

(Edge score = pair-fit χ^2)

Vertex finding

Need to determine a set of tracks to be fitted ⇒ **pattern recognition**

Traditional (**iterative**) methods: optimised for PV finding in **high pile-up**
close secondaries for **flavour tagging**

⇒ not suitable for **exotic LLP topologies**

(recently also optimised graph and GNN methods by ATLAS & CMS)

Graph-based approach: tracks are **nodes of compatibility graph** JK [arXiv:2510.00856]

Build **edges** if $\chi^2_2 \lesssim 9$

(Edge score = pair-fit χ^2)

Erase edges w/o **triplet support**

Vertex finding

Need to determine a set of tracks to be fitted ⇒ pattern recognition

Traditional (**iterative**) methods: optimised for PV finding in high pile-up
close secondaries for flavour tagging

⇒ not suitable for exotic LLP topologies

(recently also optimised graph and GNN methods by ATLAS & CMS)

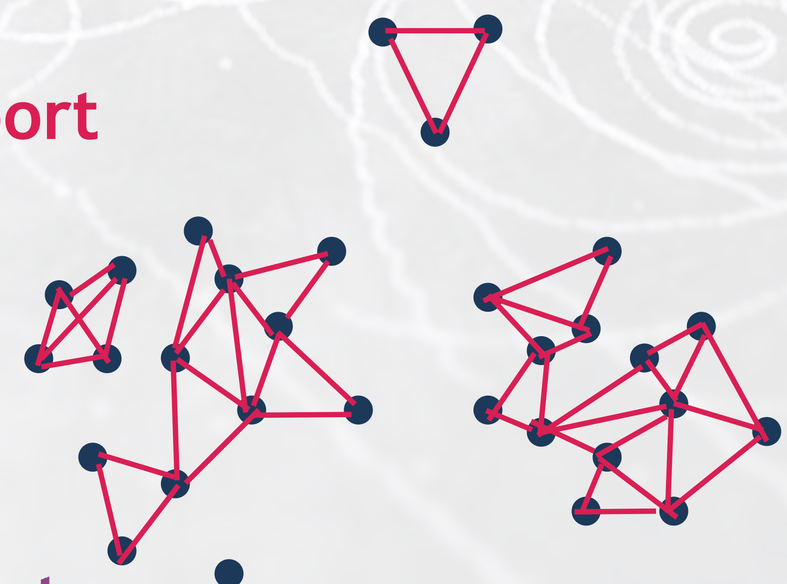
Graph-based approach: tracks are **nodes of compatibility graph**

JK [arXiv:2510.00856]

Build **edges** if $\chi^2_2 \lesssim 9$

(Edge score = pair-fit χ^2)

Erase edges w/o triplet support



More pruning/safeguards: timing consistency, mutual kNN, bridge finding/pruning, tests for intra-connectedness,…

Extract connected components

⇒ 4 vertex candidates to be fitted (+ 1 outlier track)

Vertexing performance

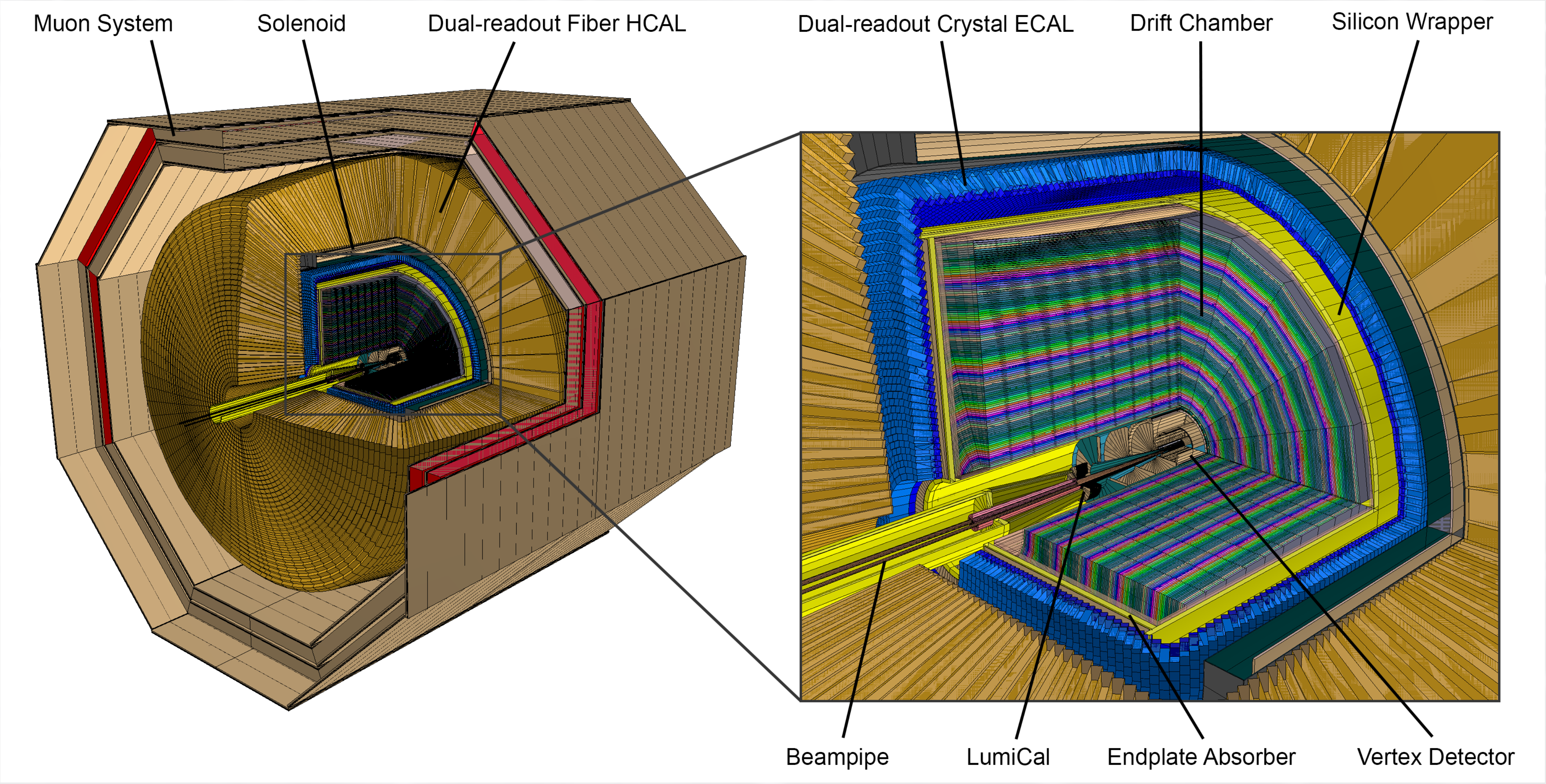
Example process at FCC-ee: $e^+e^- \to Zh, h \to NN$ with $N \to \ell^\pm q\bar{q}'$

Use **IDEA detector concept** as benchmark

Vertexing performance

Example process at FCC-ee: $e^+e^- \to Zh, h \to NN$ with $N \to \ell^\pm q\bar{q}'$

Use **IDEA detector concept** as benchmark

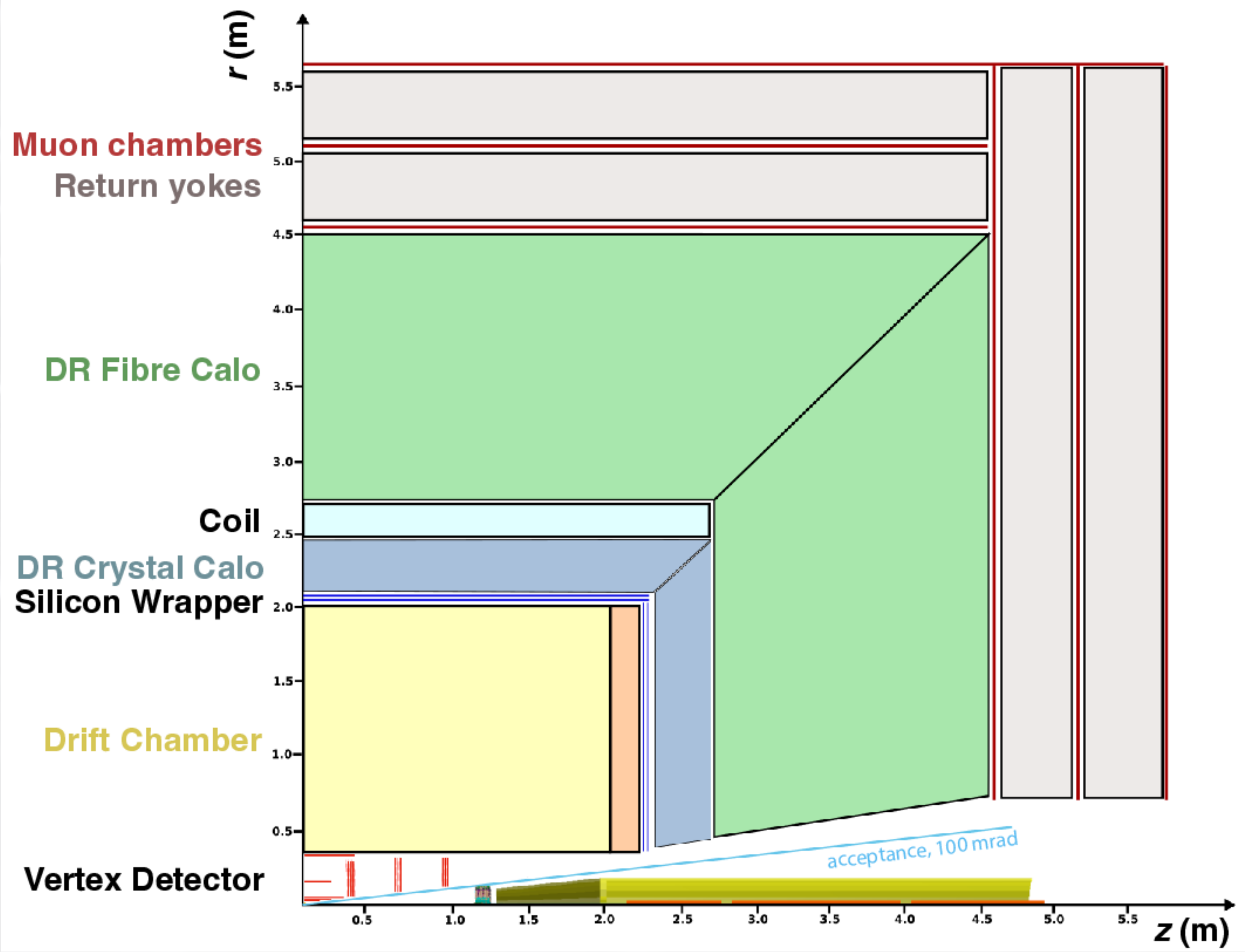


[arXiv:2502.21223]

Vertexing performance

Example process at FCC-ee: $e^+e^- \to Zh, h \to NN$ with $N \to \ell^\pm q\bar{q}'$

Use **IDEA detector concept** as benchmark



[arXiv:2502.21223]

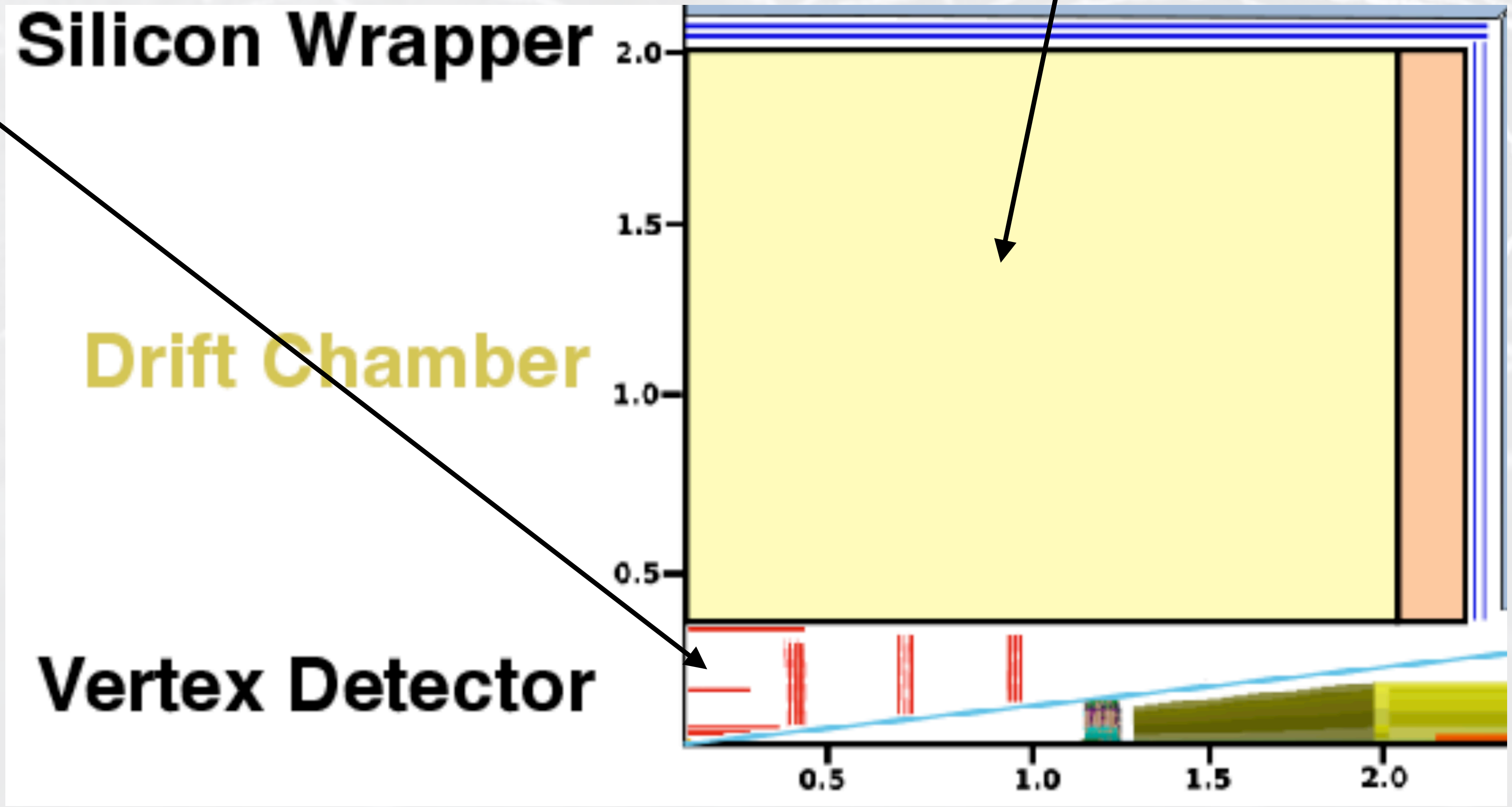
Vertexing performance

Example process at FCC-ee: $e^+e^- \to Zh, h \to NN$ with $N \to \ell^\pm q\bar{q}'$

Use **IDEA detector concept** as benchmark

Vertex Detector: single hit resolution of $3 - 7\ \mu\text{m}$, Drift Chamber: $\simeq 100\ \mu\text{m}$

Silicon wrapper: timing resolution $\simeq 30$ ps



[arXiv:2502.21223]

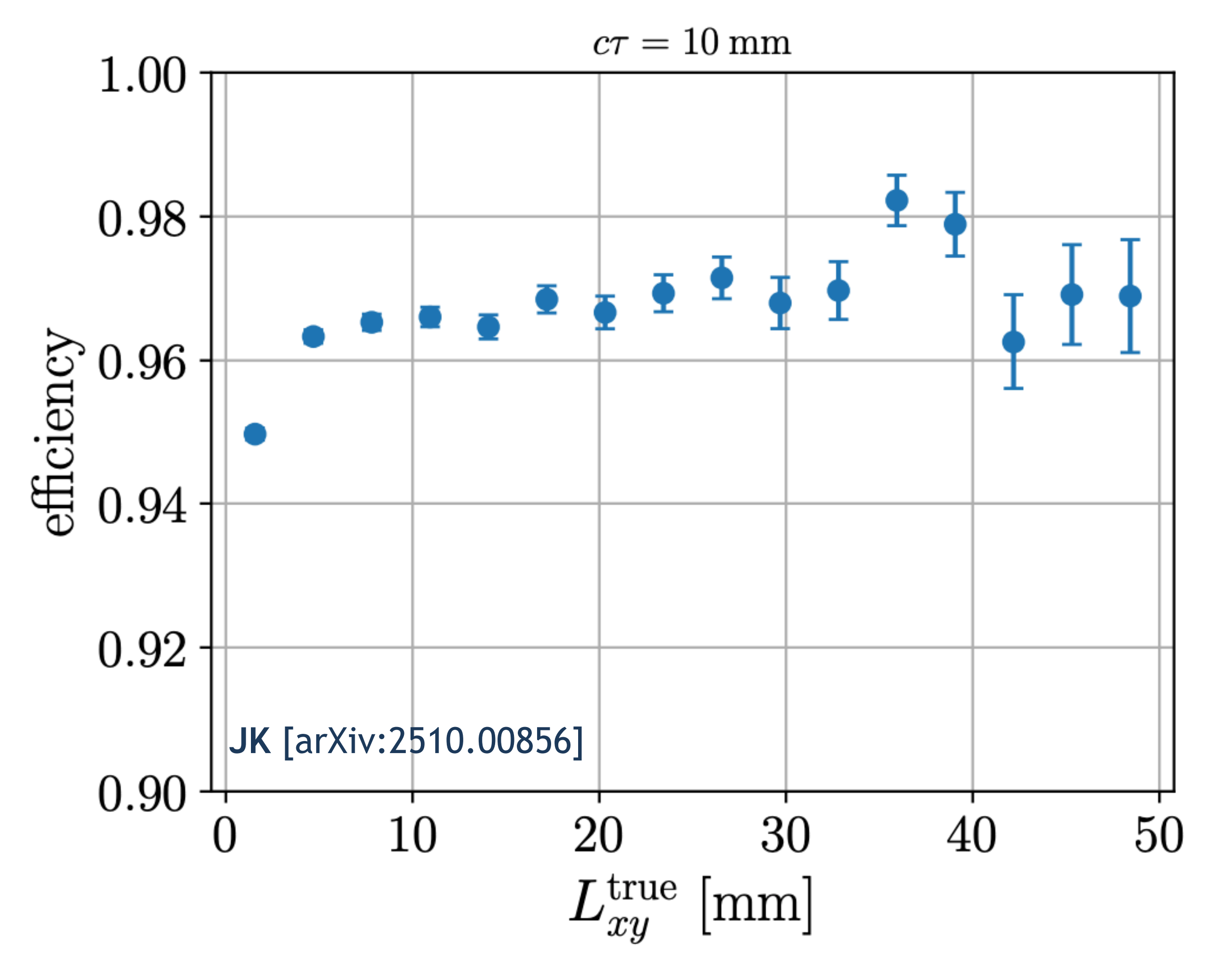
Vertexing performance

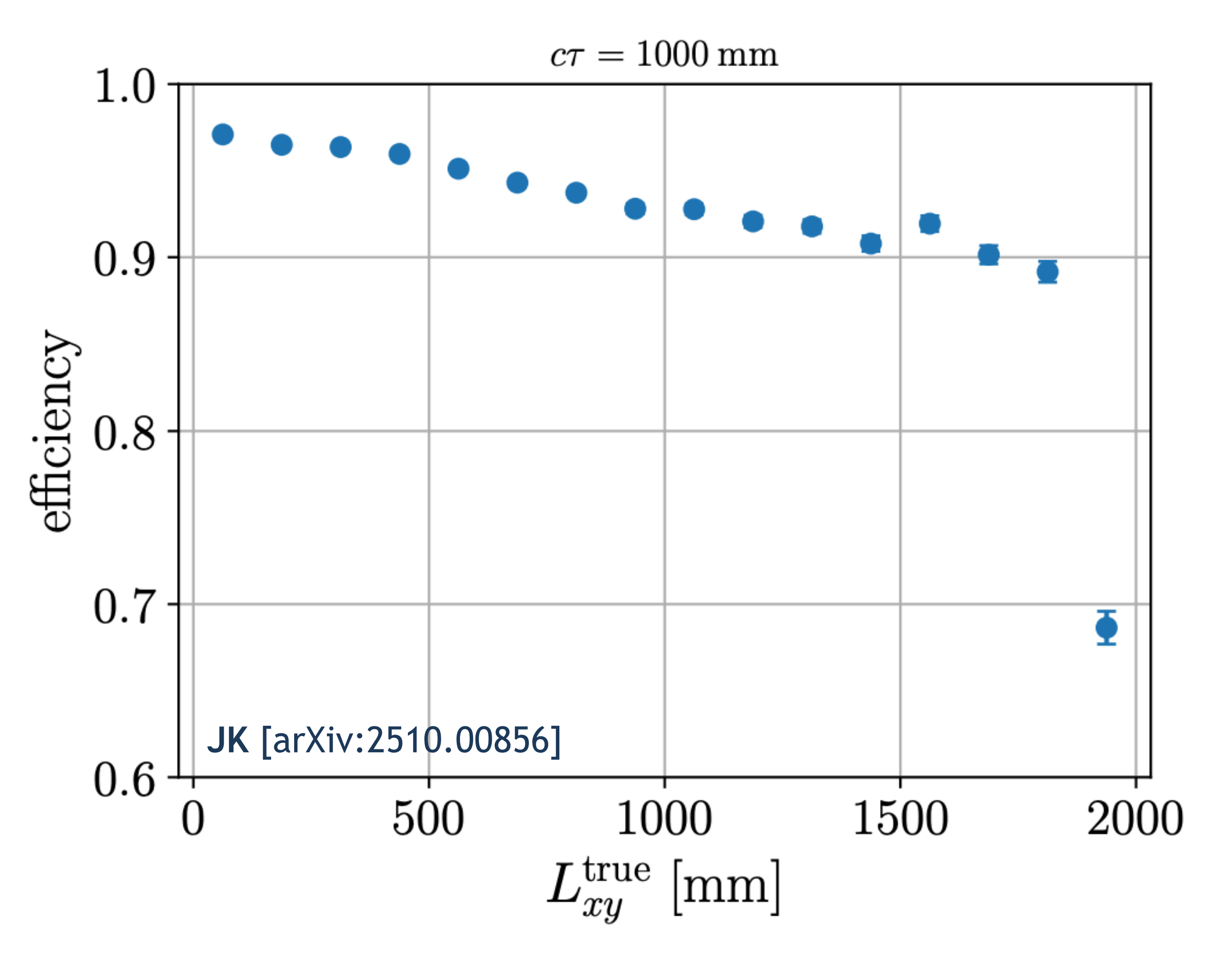
Example process at FCC-ee: $e^+e^- \to Zh, h \to NN$ with $N \to \ell^\pm q\bar{q}'$

Use **IDEA detector concept** as benchmark: fast sim. with Delphes

Delphes fast simulation

Reconstruct **two vertices simultaneously**, $m_N = 45\ \mathrm{GeV}$





Efficiency > 90% over the **entire fiducial volume**!

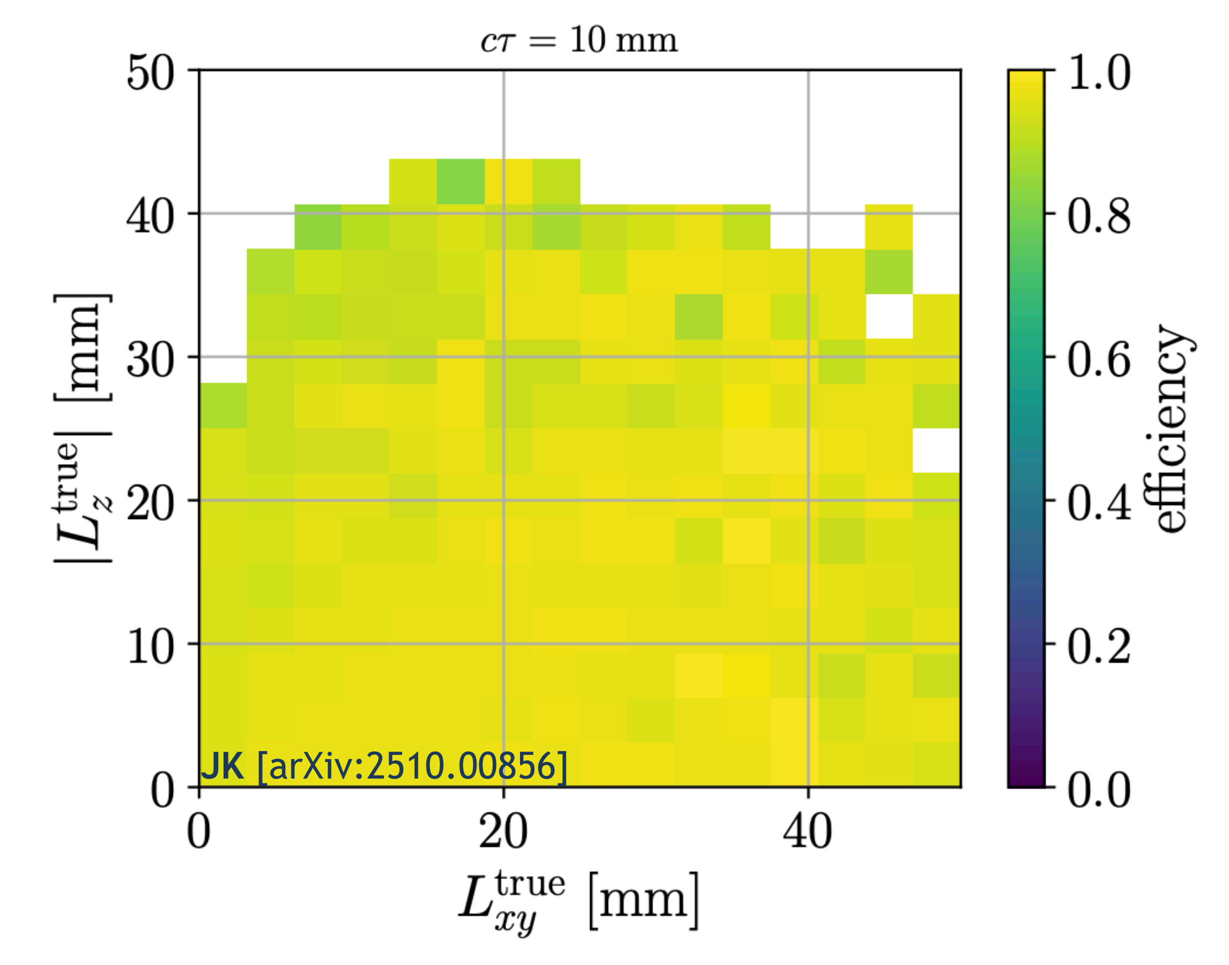
Vertexing performance

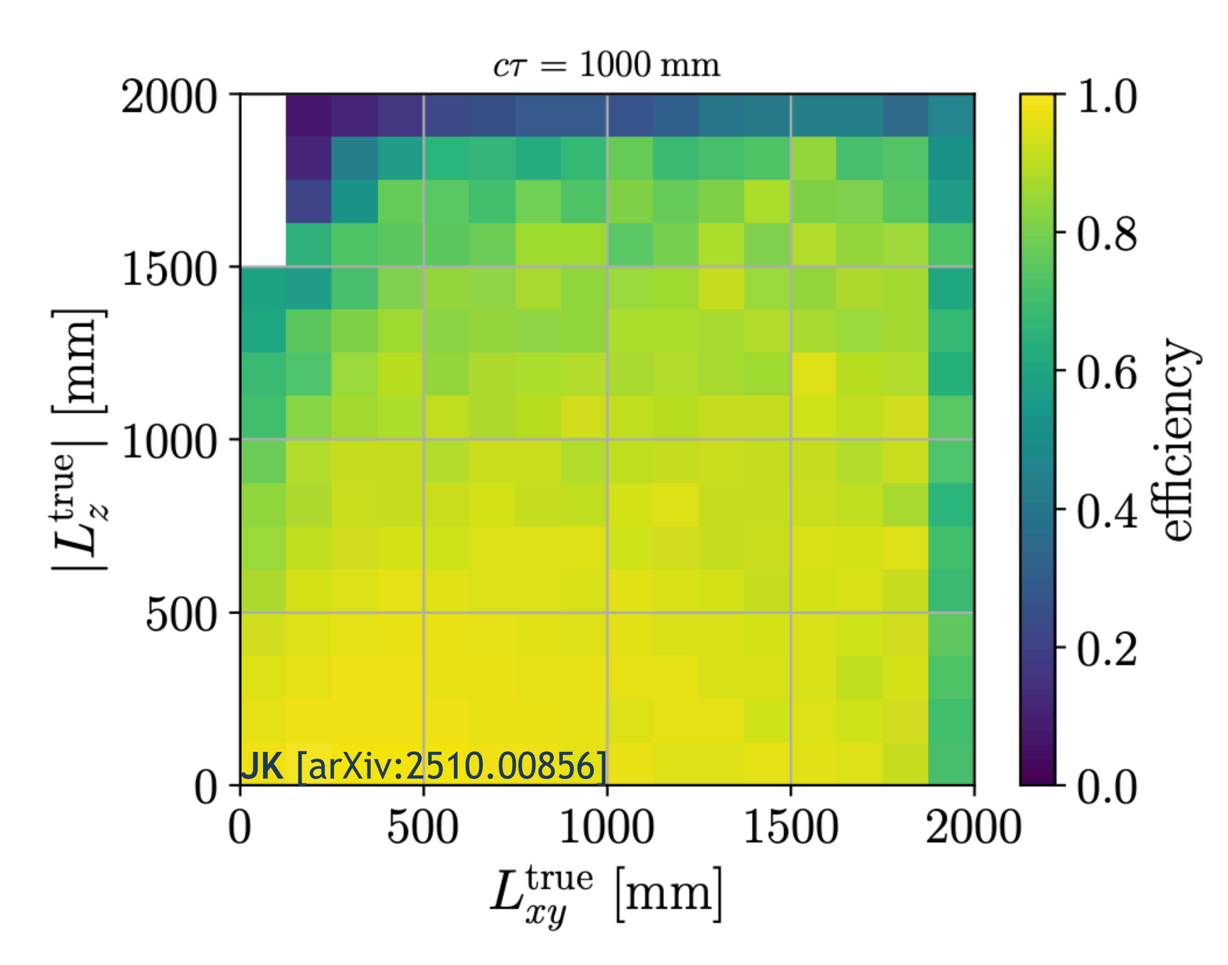
Example process at FCC-ee: $e^+e^- \to Zh, h \to NN$ with $N \to \ell^\pm q\bar{q}'$

Use **IDEA detector concept** as benchmark: fast sim. with Delphes

DELPHES
fast simulation

Reconstruct **two vertices simultaneously,** $m_N = 45\ \mathrm{GeV}$





Efficiency > 90% over the **entire fiducial volume**!

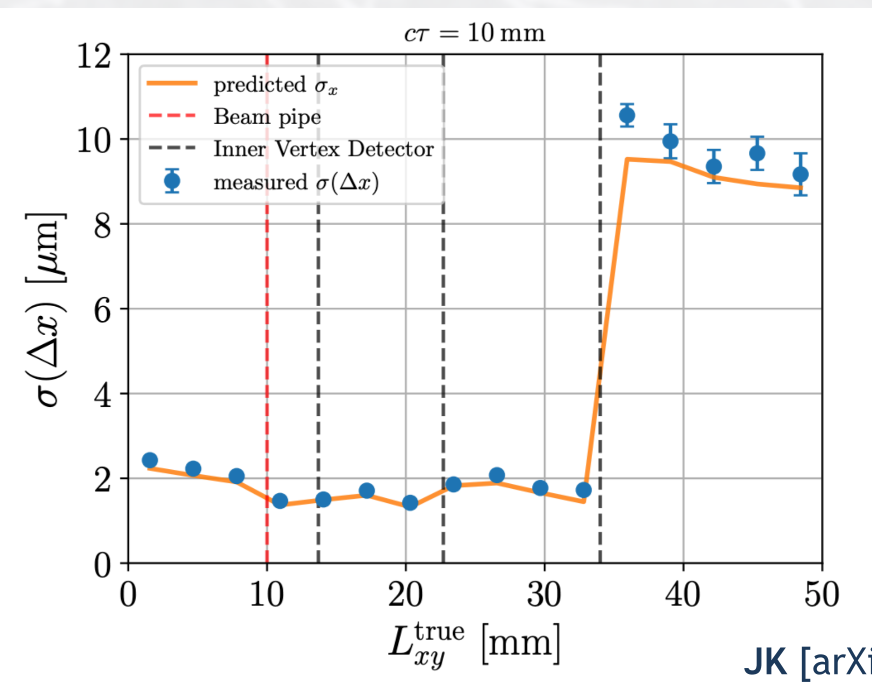
Vertexing performance

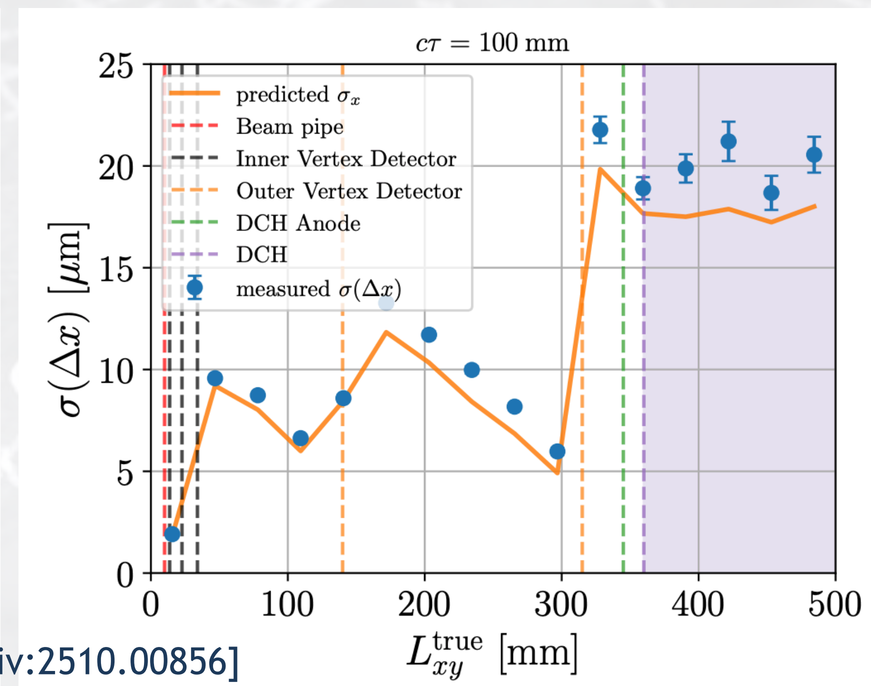
Example process at FCC-ee: $e^+e^- \to Zh, h \to NN$ with $N \to \ell^\pm q\bar{q}'$

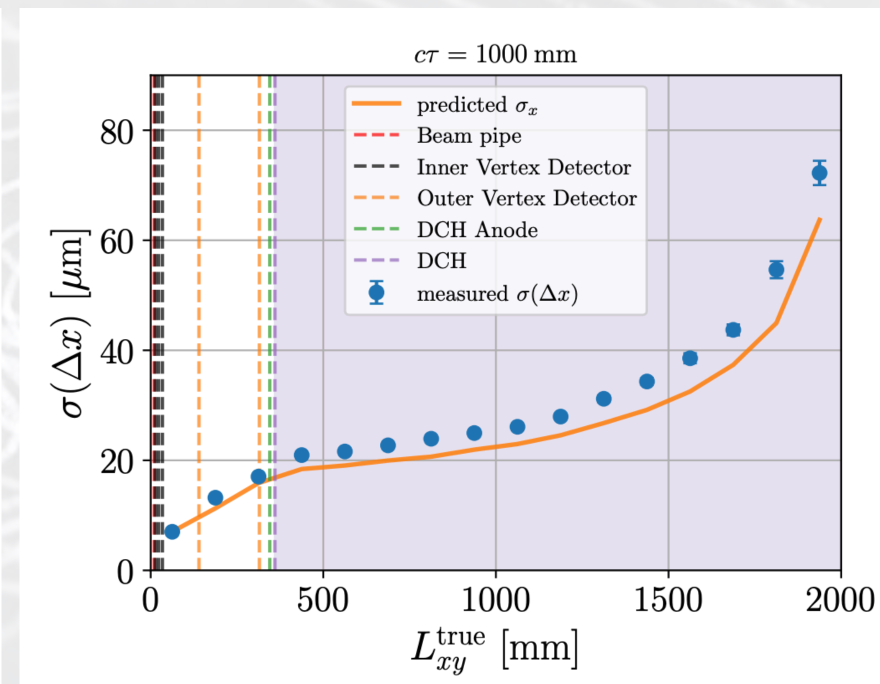
Use **IDEA detector concept** as benchmark: fast sim. with Delphes

DELPHES
fast simulation

Reconstruct **two vertices simultaneously**, $m_N = 45\ \mathrm{GeV}$







JK [arXiv:2510.00856]

Resolution $\lesssim 2\,\mu\mathrm{m}$ in the inner vertex Detector!

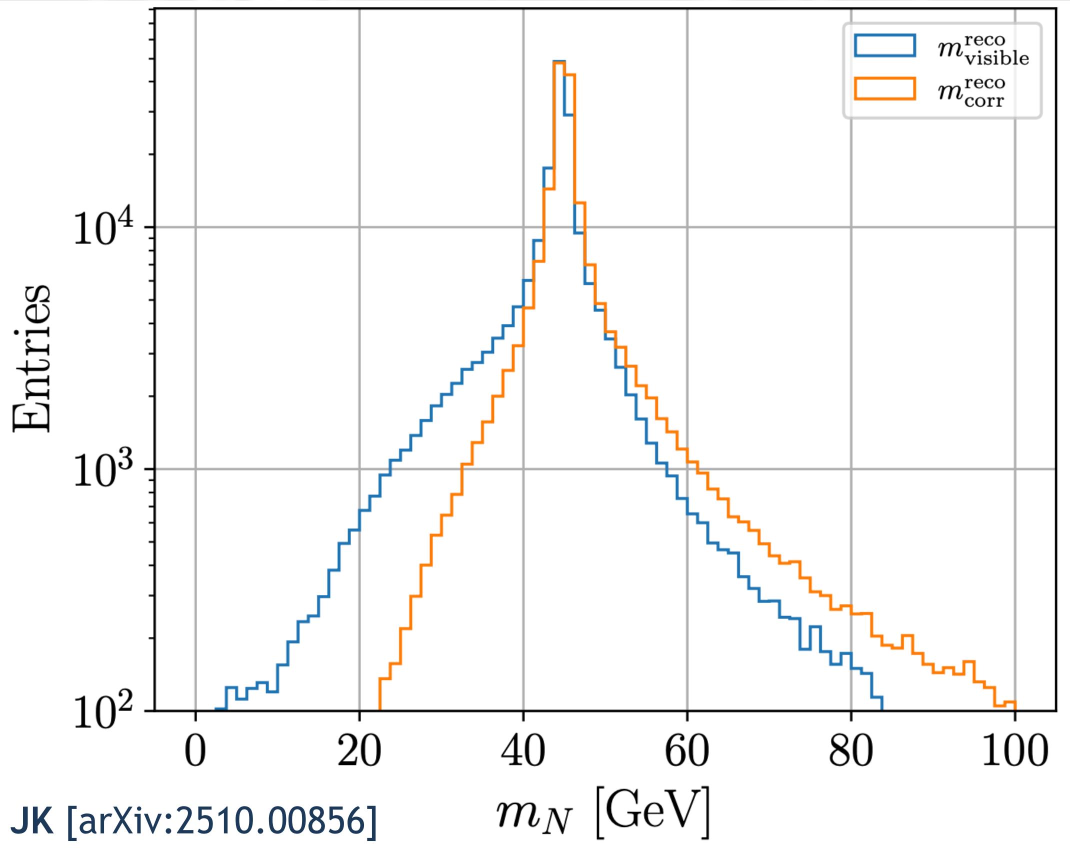
Uncertainty model agrees well with MC-data (within 5-10%)

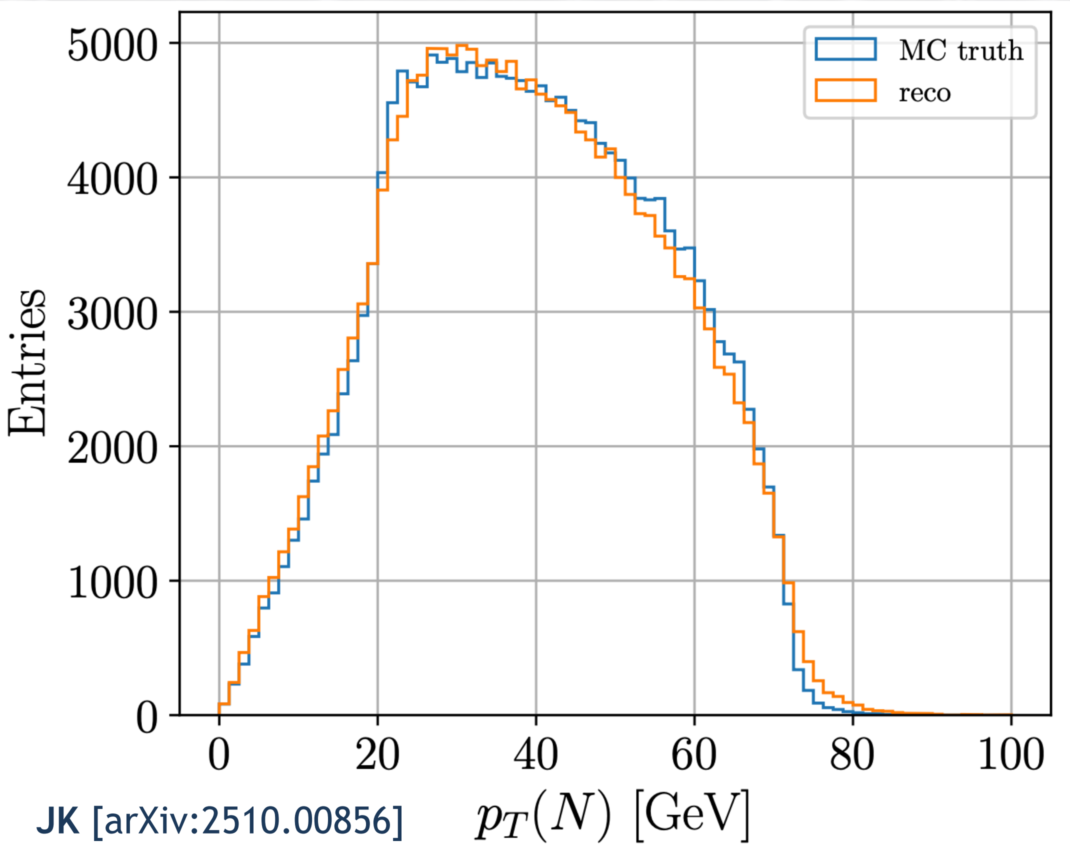
Long-lived Particle reconstruction

Match reconstructed **vertices** to **leptons** and **jets** to reconstruct **LLP kinematics**

Define p_T –weighted track overlap score:

$$s_{\text{overlap}} = \frac{2\sum_{i\in\text{shared}}(p_T^i)^\alpha}{\sum_{i\in\text{jet}}(p_T^i)^\alpha + \sum_{i\in\text{DV}}(p_T^i)^\alpha} \quad \Rightarrow \quad p^\mu_{\text{LLP}} = \sum_{i\in\text{jets, leptons}} p_i^\mu$$





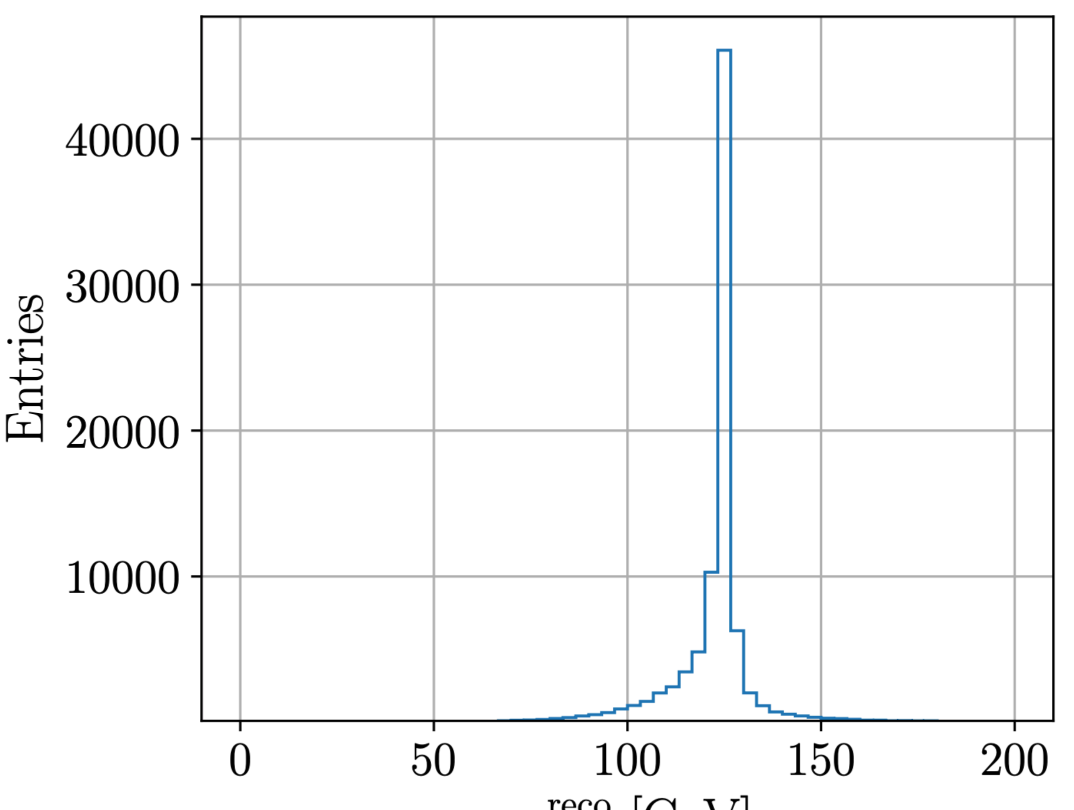
2 mass proxies: p^2_{LLP} & $m_{\text{corr}} = \sqrt{m^2_{\text{vis}} + p^2_\perp} + p_\perp$

Very good agreement of reconstructed p_T

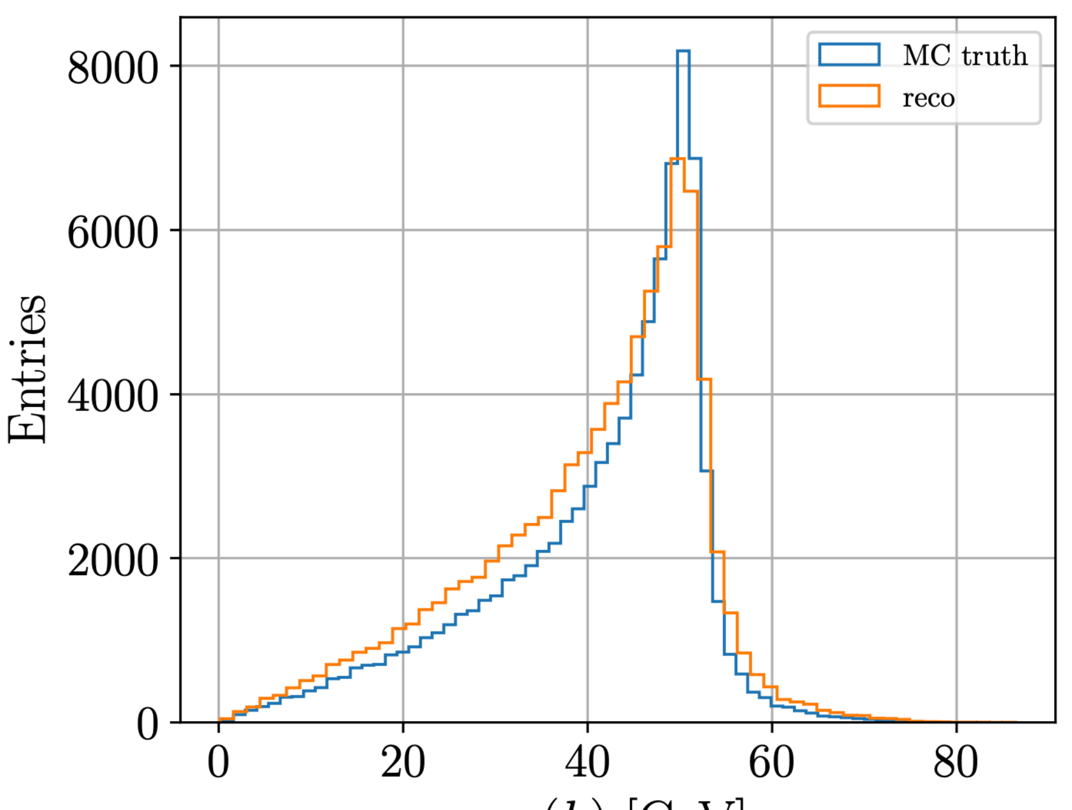
Long-lived Particle reconstruction

Match reconstructed **vertices** to **leptons** and **jets** to reconstruct **LLP kinematics**

Reconstruct **parent kinematics** ($e^+e^- \to Zh, h \to NN$)



JK [arXiv:2510.00856]



JK [arXiv:2510.00856]

$\Rightarrow$ Allows to reconstruct the entire production & decay topology!

Full analysis

Example process at FCC-ee: $e^+e^- \to Zh, h \to NN$ with $N \to \ell^\pm q\bar{q}'$

Scan over wide range of masses and lifetimes m_N & $c\tau_N$

Event (pre-)selection:

Possible SM backgrounds: heavy flavour (b , c hadrons, τ-decays)

- **4 reconstructed jets** (exclusive durham k_T, remove isolated leptons from jets)
- Reconstructed **primary vertex** with ≤ 2 (lepton) tracks (from $Z \to \ell^+\ell^-, \nu\bar{\nu}$)
- 2 **reconstructed vertices** with at least 3 tracks each
- **Transverse displacement** $L_{xy} \geq 500\ \mu\text{m}$

Mitigates most V_0 backgrounds & material interactions

Full analysis

Example process at FCC-ee: $e^+e^- \to Zh, h \to NN$ with $N \to \ell^\pm q\bar{q}'$

Scan over wide range of masses and lifetimes m_N & $c\tau_N$

Event (pre-)selection:

Possible SM backgrounds: heavy flavour (b , c hadrons, τ-decays)

- **4 reconstructed jets** (exclusive durham k_T, remove isolated leptons from jets)
- Reconstructed **primary vertex** with ≤ 2 (lepton) tracks (from $Z \to \ell^+\ell^-, \nu\bar{\nu}$)
- 2 **reconstructed vertices** with at least 3 tracks each
- **Transverse displacement** $L_{xy} \geq 500\ \mu\mathrm{m}$

Mitigates most V_0 backgrounds & material interactions

Additional kinematic cuts: $\Rightarrow$ reduce SM backgrounds to < 1 Event

- Window cut on $m_{\mathrm{corr}} \in [0.8\, m_N, 1.5\, m_N]$
- Window cut on $m_h^{\mathrm{reco}} \in [100\, , 150]\ \mathrm{GeV}$

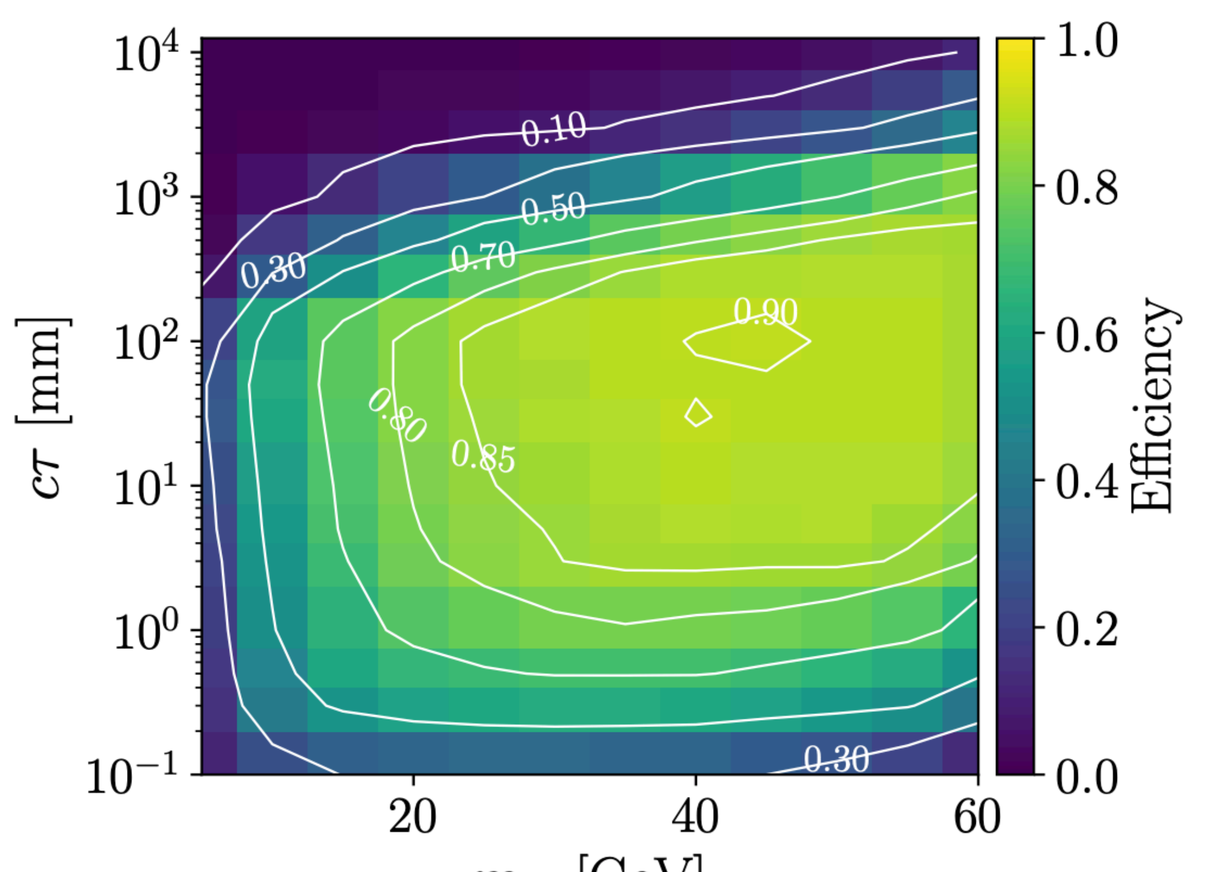
Remaining (rare) backgrounds: γ conversions, miss-ID, BIB, cosmic rays, (can all be vetoed/mitigated)

Full analysis

Example process at FCC-ee: $e^+e^- \to Zh, h \to NN$ with $N \to \ell^\pm q\bar{q}'$

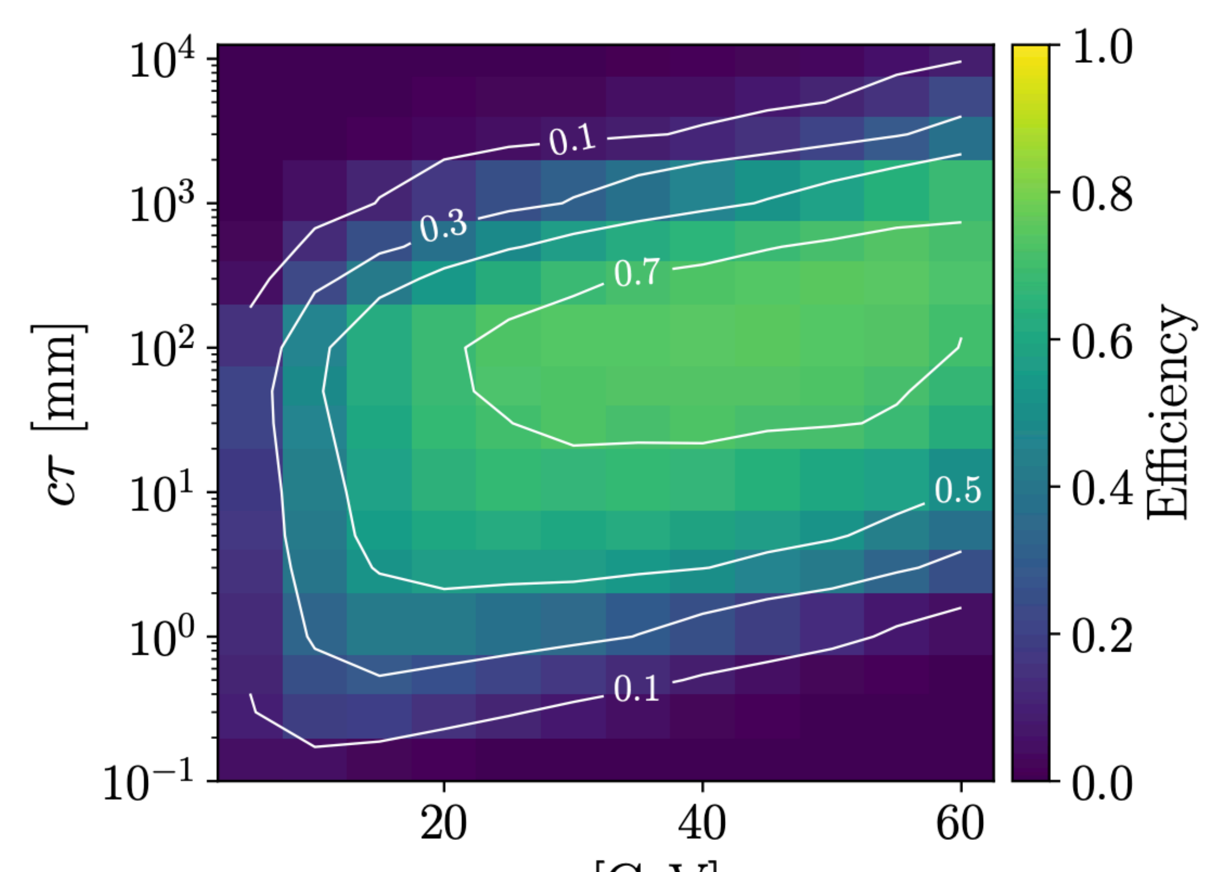
Scan over wide range of masses and lifetimes m_N & $c\tau_N$

Event (pre-)selection:



JK [arXiv:2510.00856]

Additional kinematic cuts:

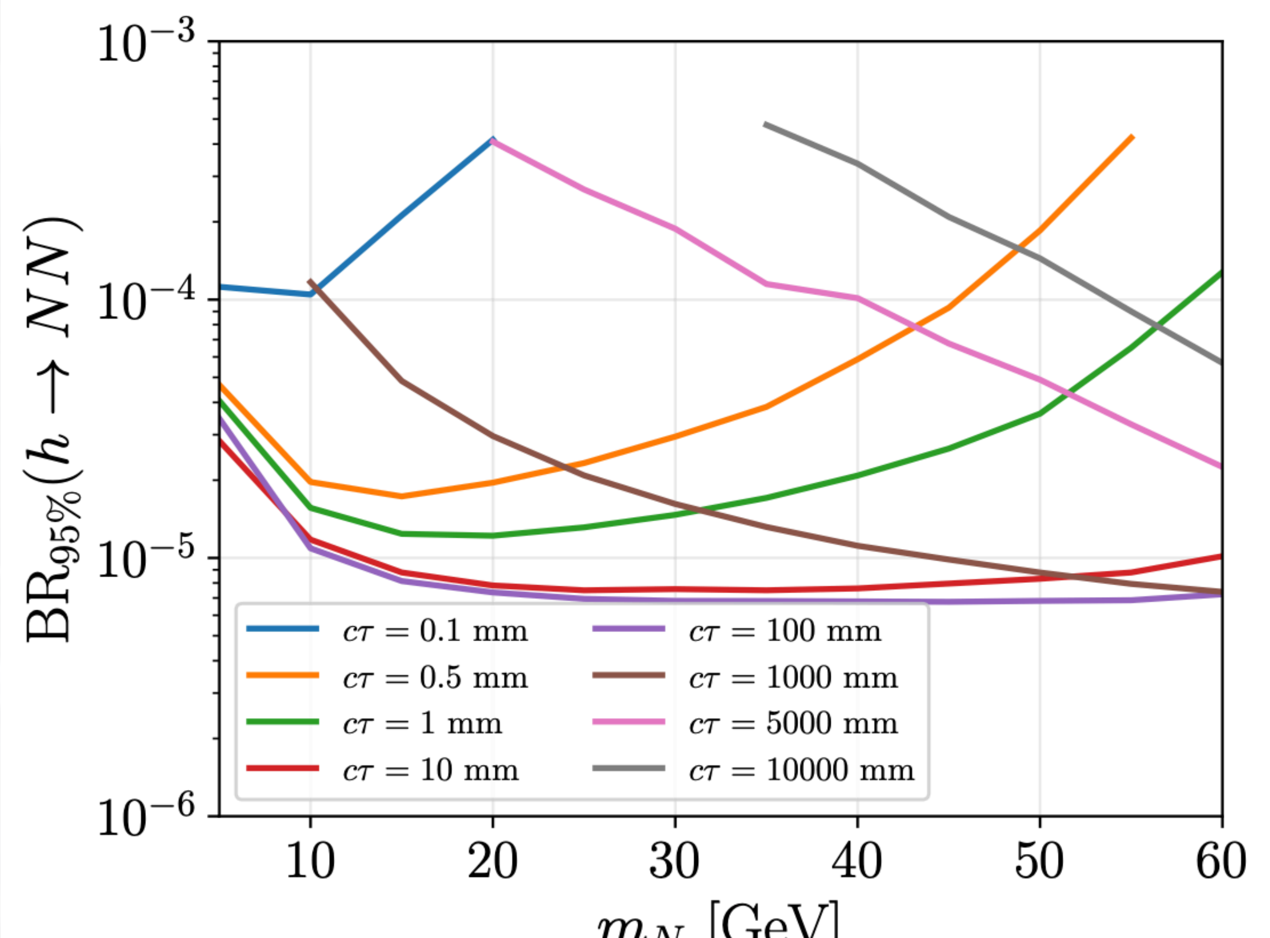


JK [arXiv:2510.00856]

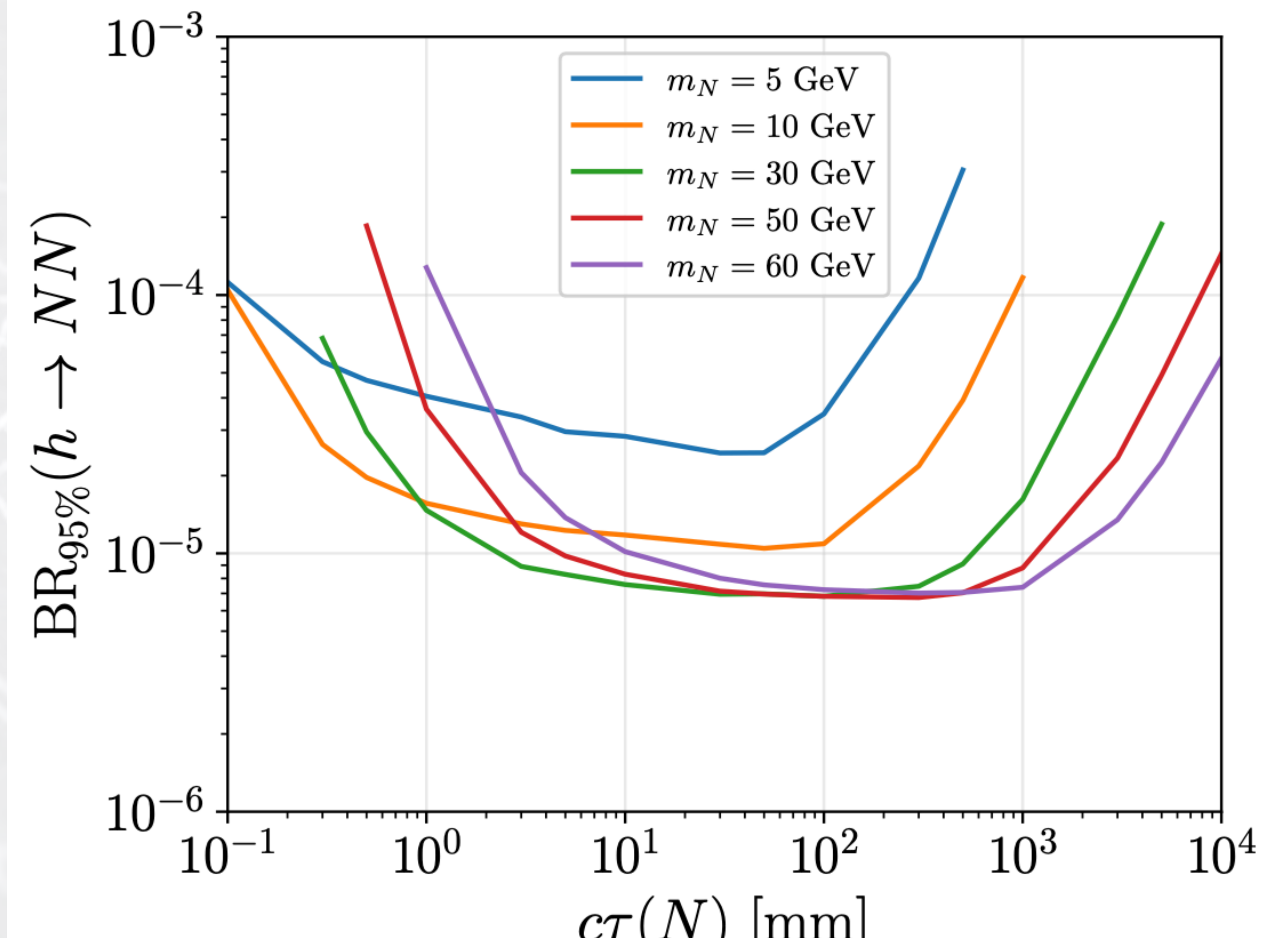
$\Rightarrow$ very high **selection/reconstruction efficiencies** over wide range of m_N & $c\tau_N$

Results: upper limits on $\mathrm{BR}(h \to NN)$

Set upper limits at $95\,\%$ CL: at least 3 reconstructed events



JK [arXiv:2510.00856]



JK [arXiv:2510.00856]

$\Rightarrow$ FCC-ee sensitive to $\mathrm{BR}(h \to NN) \lesssim 6 - 7 \times 10^{-6}$

Conclusions

- **Long-lived particles** common to many **BSM theories**
  - $\Rightarrow$ Rich hep-ph and hep-ex programs
  - $\Rightarrow$ FCC-ee strongly sensitive to many interesting signatures
- **No suitable tools for pheno available :(**

Conclusions

- Long-lived particles common to many **BSM theories**

  ⇒ Rich hep-ph and hep-ex programs

  ⇒ FCC-ee strongly sensitive to many interesting signatures

- ~~No suitable tools for pheno available~~

- **Developed LLP-dedicated vertex fitting and vertex finding algorithms**

Entire track-to-LLP-kinematics pipeline implemented in

Delphes (plug & play)

Check it out: https://github.com/jkriewald/delphes-LLP

Conclusions

- Long-lived particles common to many **BSM theories**
  - ⇒ Rich hep-ph and hep-ex programs
  - ⇒ FCC-ee strongly sensitive to many interesting signatures

- ~~No suitable tools for pheno available~~

- **Developed LLP-dedicated vertex fitting and vertex finding algorithms**

Entire track-to-LLP-kinematics pipeline implemented in

Delphes (plug & play)

Check it out: https://github.com/jkriewald/del

Thanks for listening!

Bonus content

Weighted fit (IRLS)

Assign (sigmoid) weights based on single track χ^2

$$w_i = \psi(\chi_i^2) = \frac{1}{1 + \exp(\frac{\chi_i^2 - \chi_c^2}{\beta})}$$

Minimise weighted likelihood:

$$F_w(\{s_i\}, \vec{v}) = \frac{1}{2}\left[\sum_i^{N_{\text{tracks}}} w_i(\vec{v} - \vec{x}_i(s_i))^T\, W_i(s)\,(\vec{v} - \vec{x}_i(s_i))\right]$$

Outlier weights $\rightarrow 0$, inlier weights $0.5 \leq w_i \leq 1$

Vertex timing

Simplified track timing model: $$\tau_i(s_i) = \tau_i^{\mathrm{ref}} - \frac{s_i^{\mathrm{ref}} - s_i}{\beta_i}$$

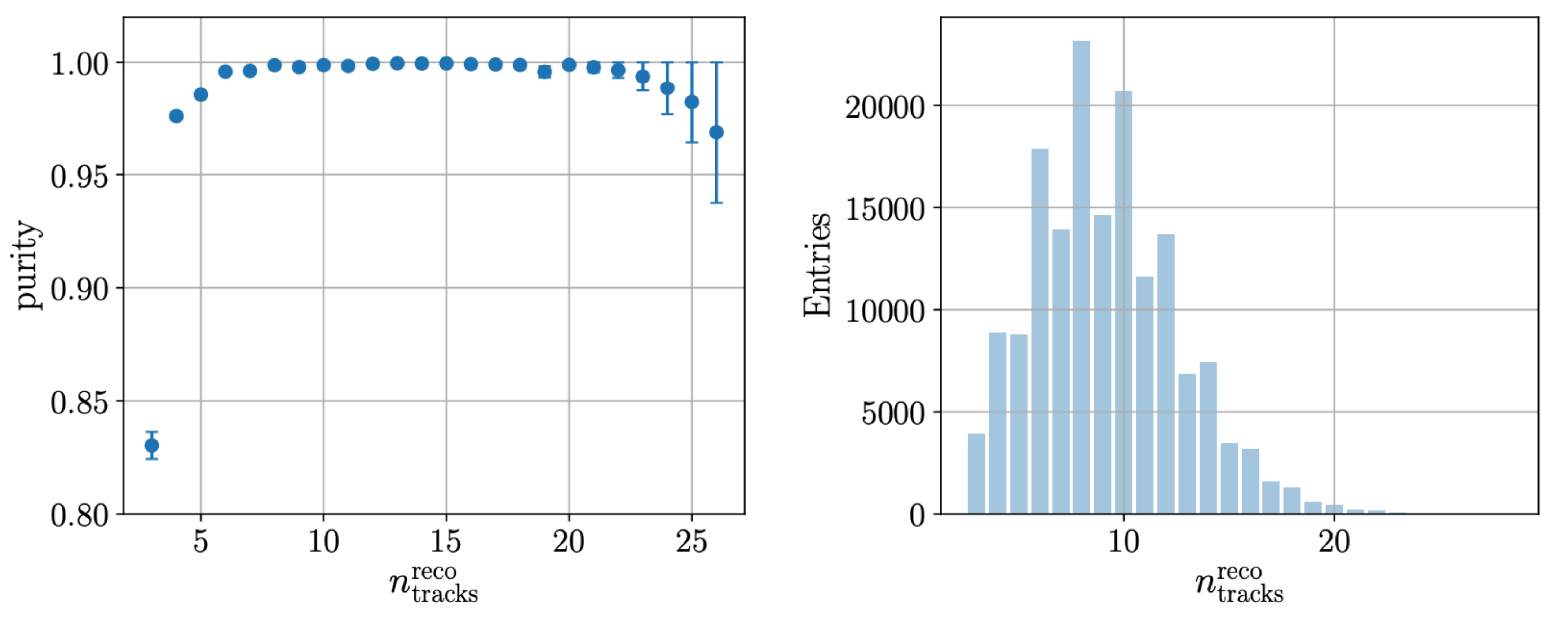
Add to likelihood:

$$F_\tau(\{s_i\}, \tau_v) = \frac{1}{2} \sum_i \frac{(\tau_v - \tau_i(s_i))^2}{(\sigma_\tau^i)^2}$$

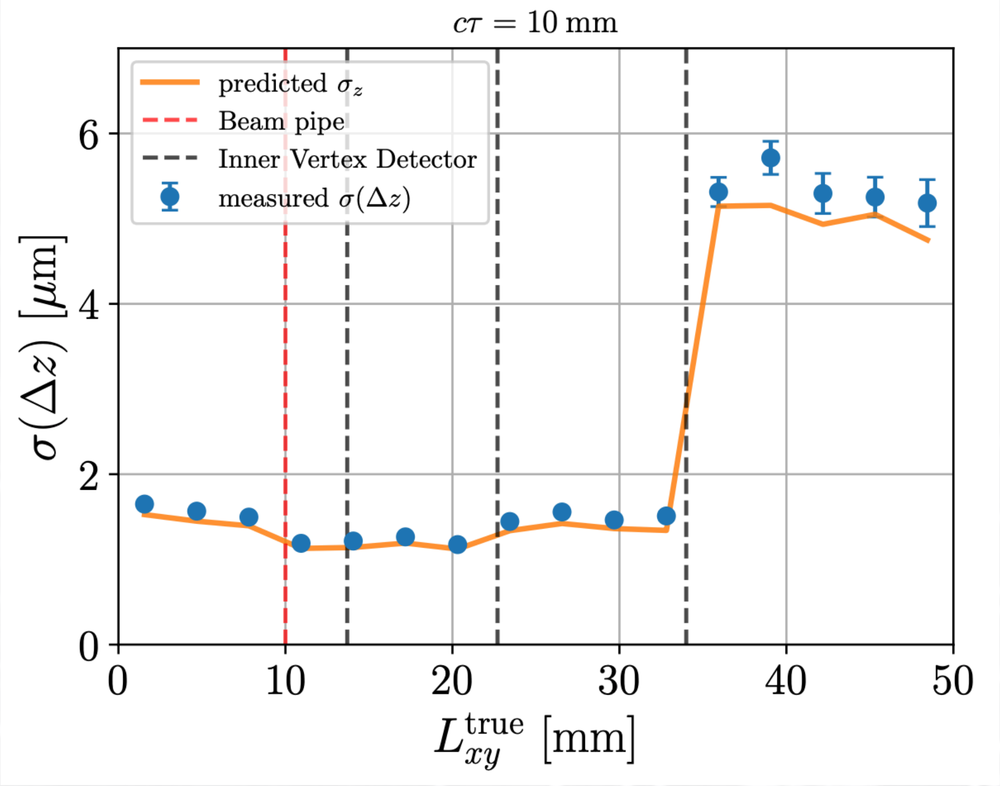
(Approximately) decoupled from spatial fit: ($\sigma_\tau \gg \sigma_{\vec{x}}$)

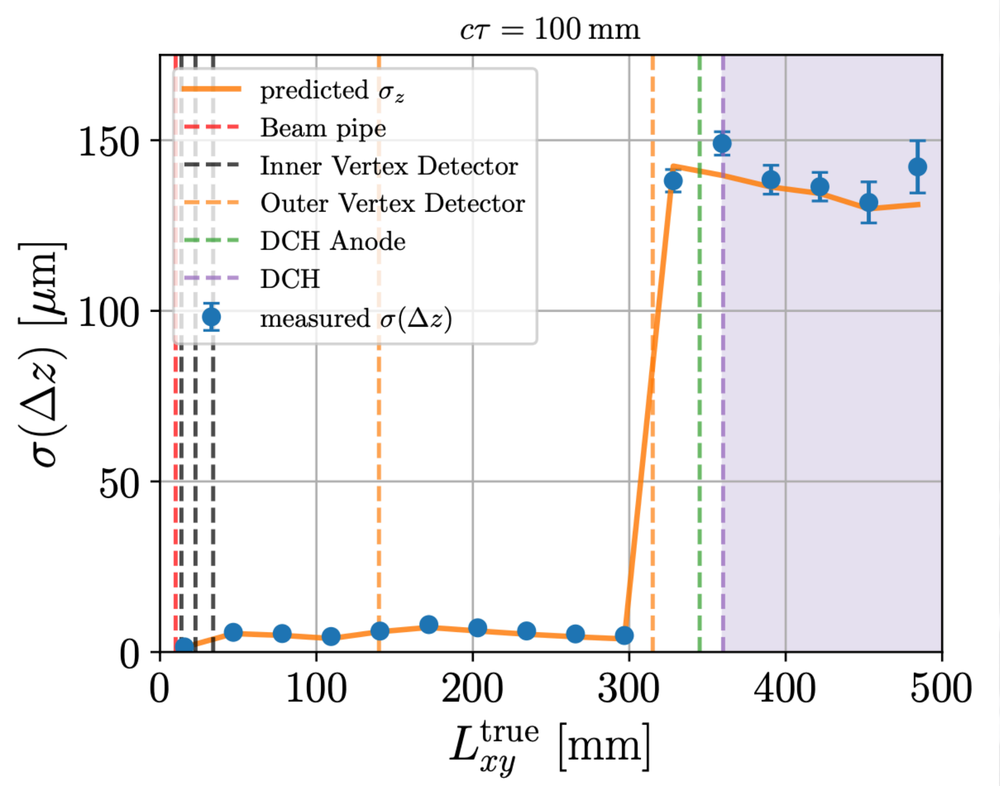
$$\tau_v^\star = \frac{1}{\sum_i \frac{1}{(\sigma_\tau^i)^2}} \sum_j \frac{1}{(\sigma_\tau^j)^2} \tau_j(s_j^\star)$$

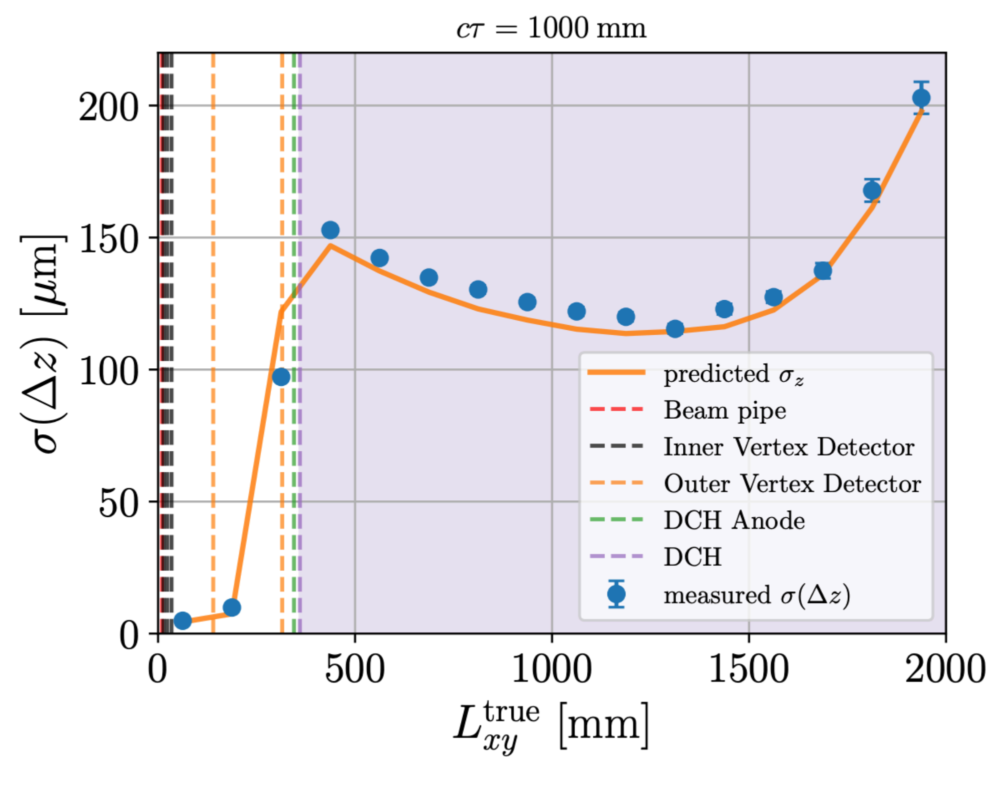
Purity


purity
1.00
0.95
0.90
0.85
0.80
5
10
15
20
25
$n_{\mathrm{tracks}}^{\mathrm{reco}}$
Entries
20000
15000
10000
5000
0
10
20
$n_{\mathrm{tracks}}^{\mathrm{reco}}$

Longitudinal resolution







Pseudo-rapidity reconstruction