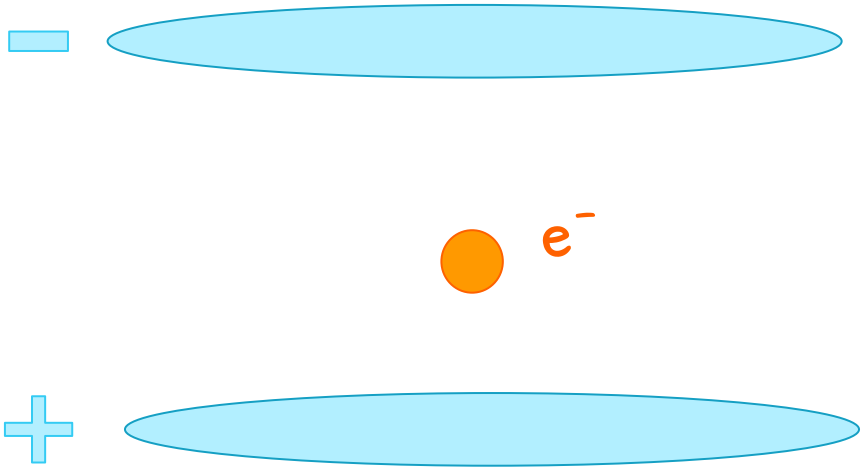
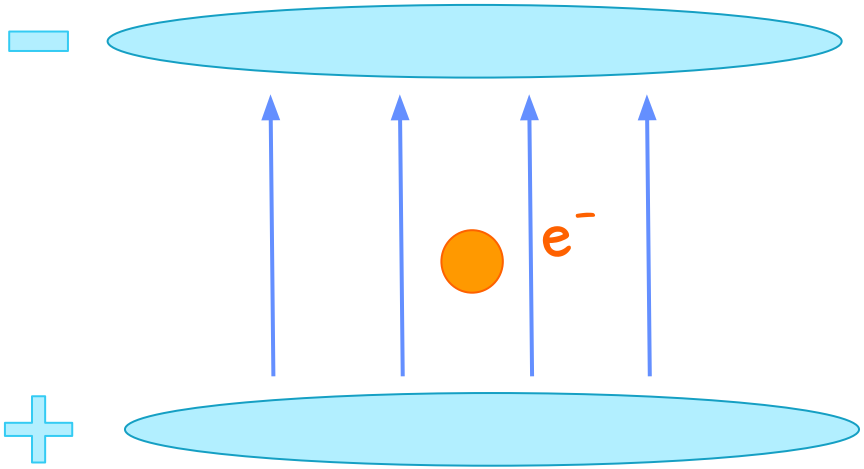
Determining a TPC Electric Field using Physics-Informed Continuous Normalizing Flows

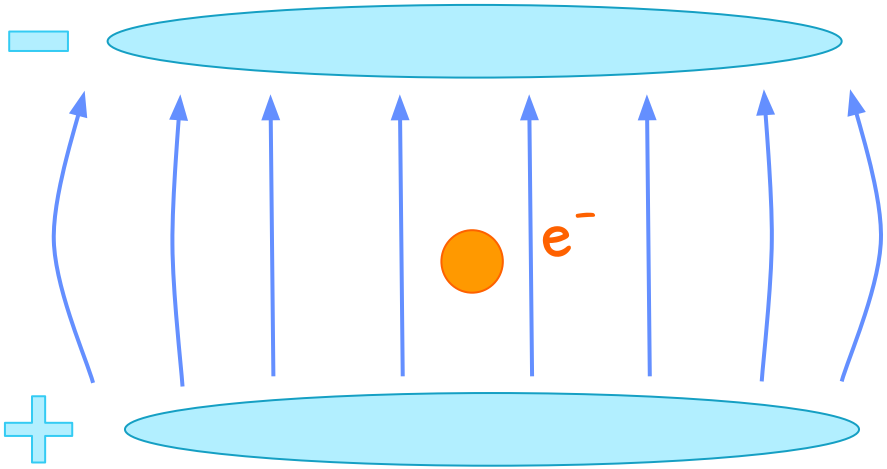
Ivy Li[1] (il11@rice.edu), Peter Gaemers[2], Juehang Qin[1], Naija Bruckner[1]

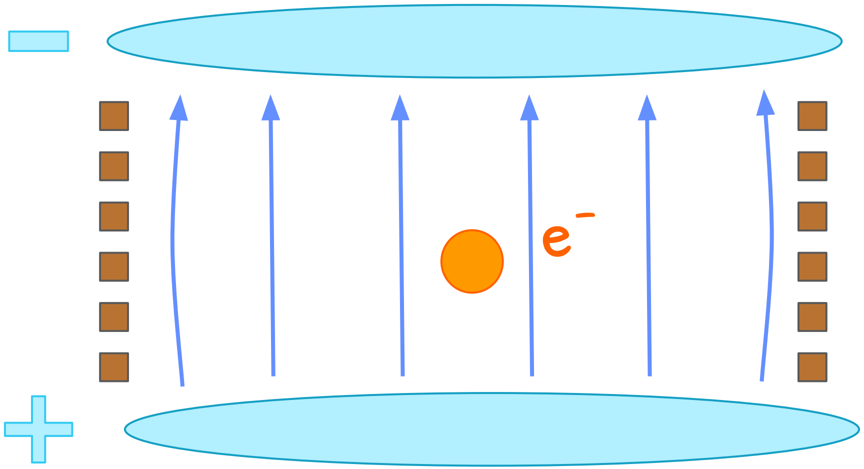
Oct. 30, 2025
NPML 2025 - University of Tokyo

[1]Rice University, [2]SLAC National Accelerator Laboratory

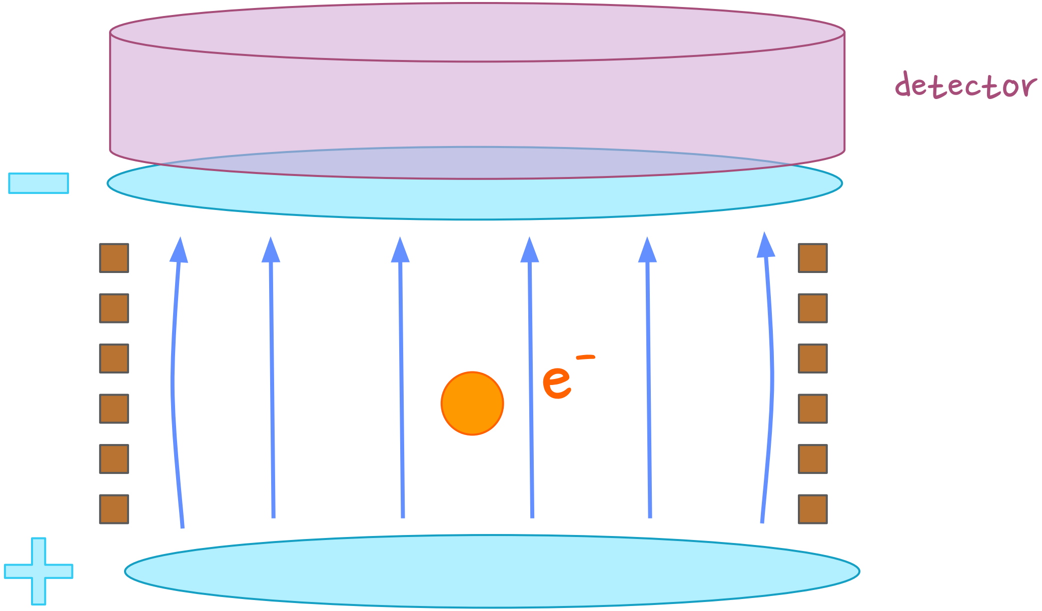

−
e^-
+

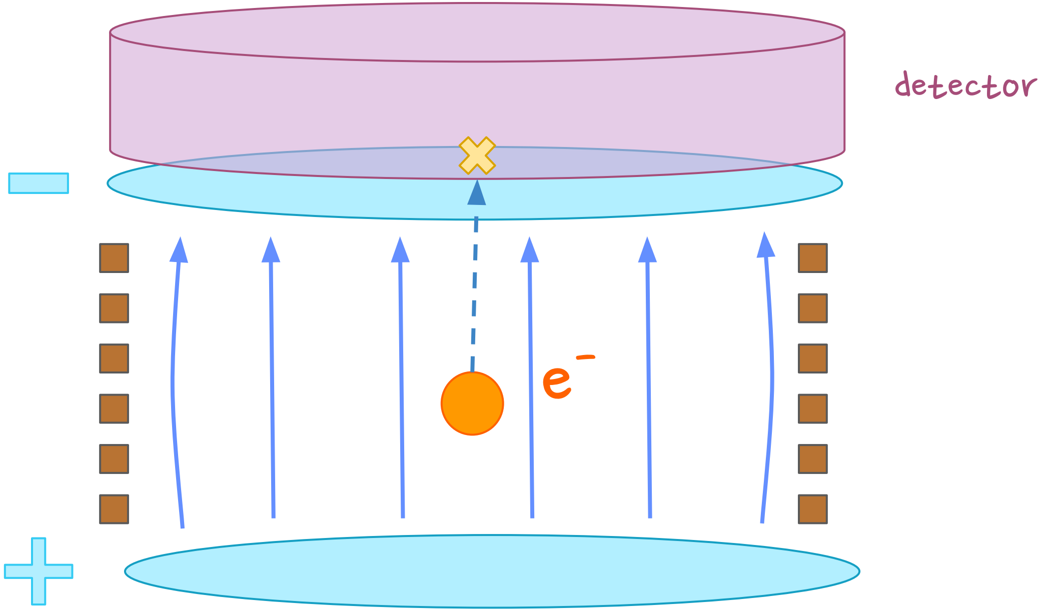

e^-

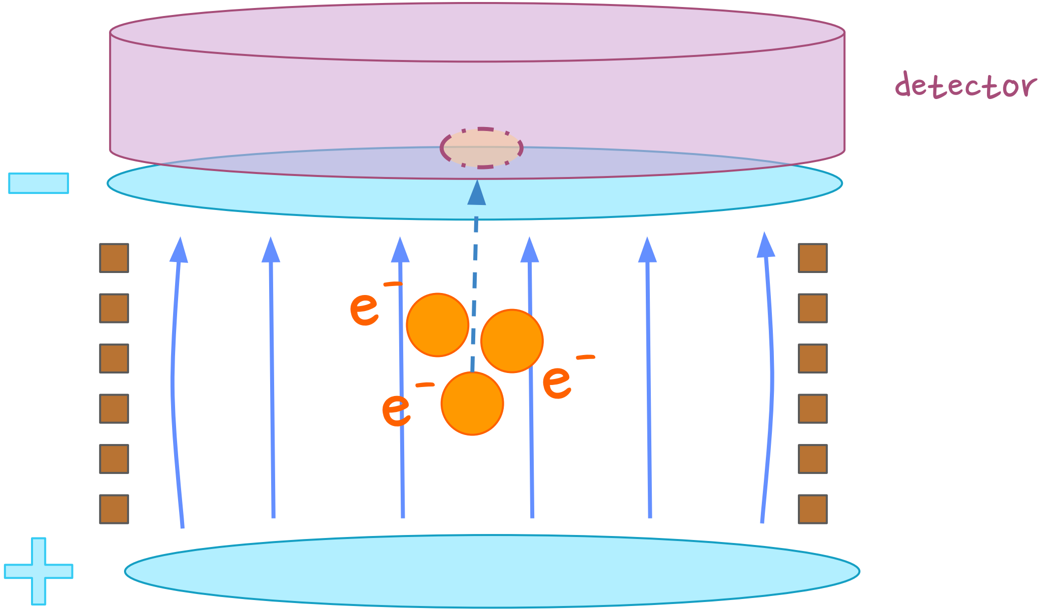

e^-

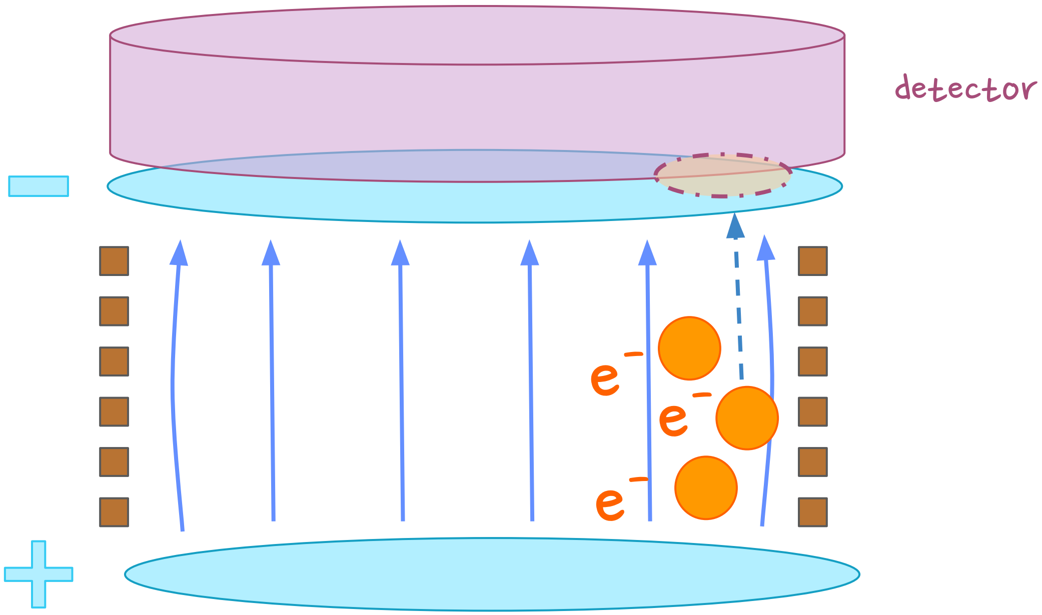


For XENONnT field shaping rings and geometry, see [Eur. Phys. J. C 84, 138 (2024)]

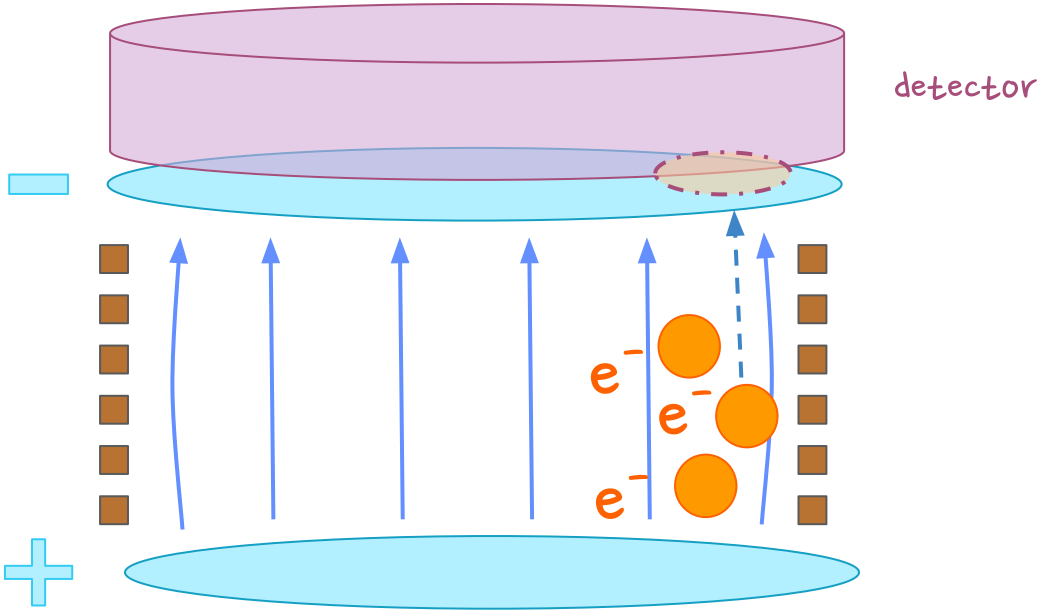

detector
e⁻


detector
e^-

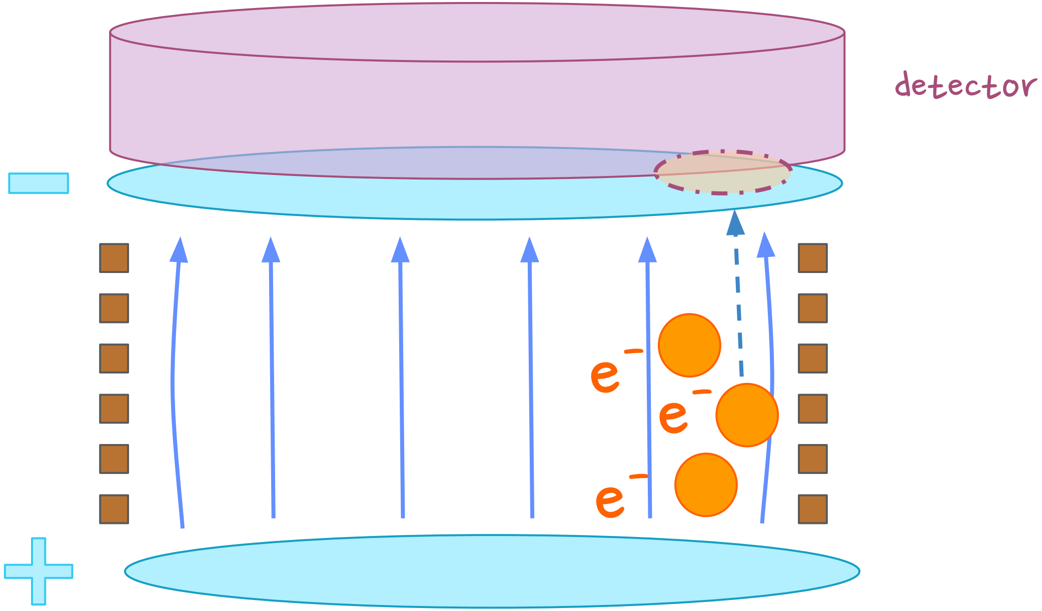

detector
e⁻
e⁻
e⁻


detector
e⁻
e⁻
e⁻
−
+

I know where my electrons ended up (sort of).

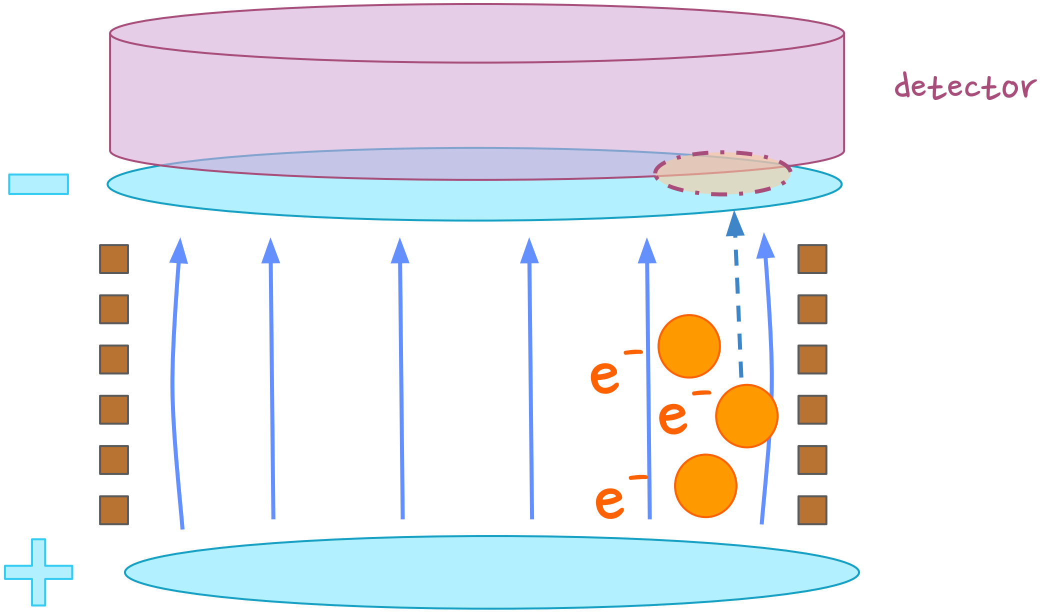


I know where my electrons ended up (sort of).

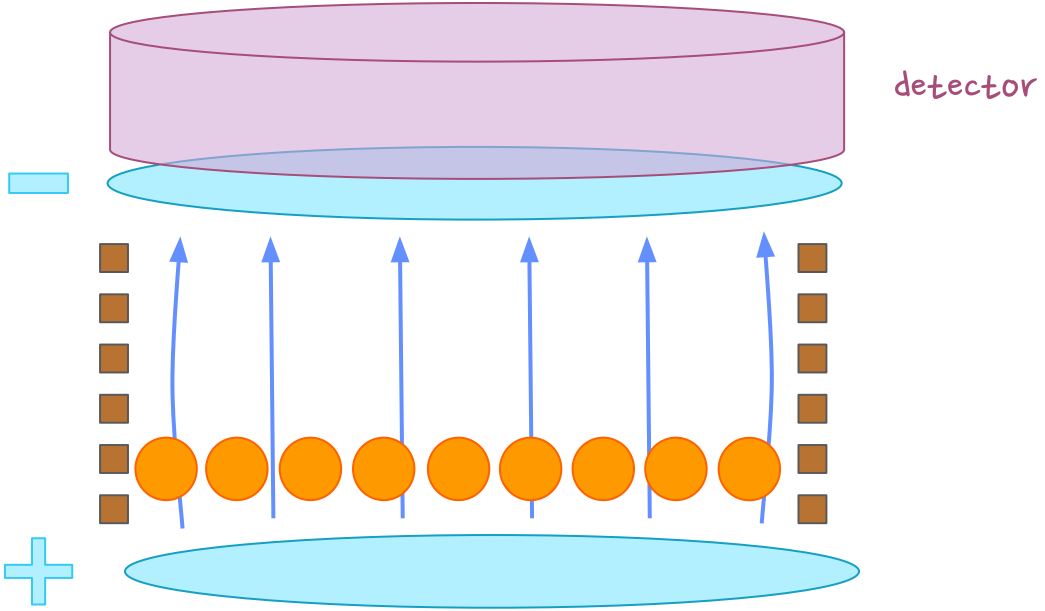


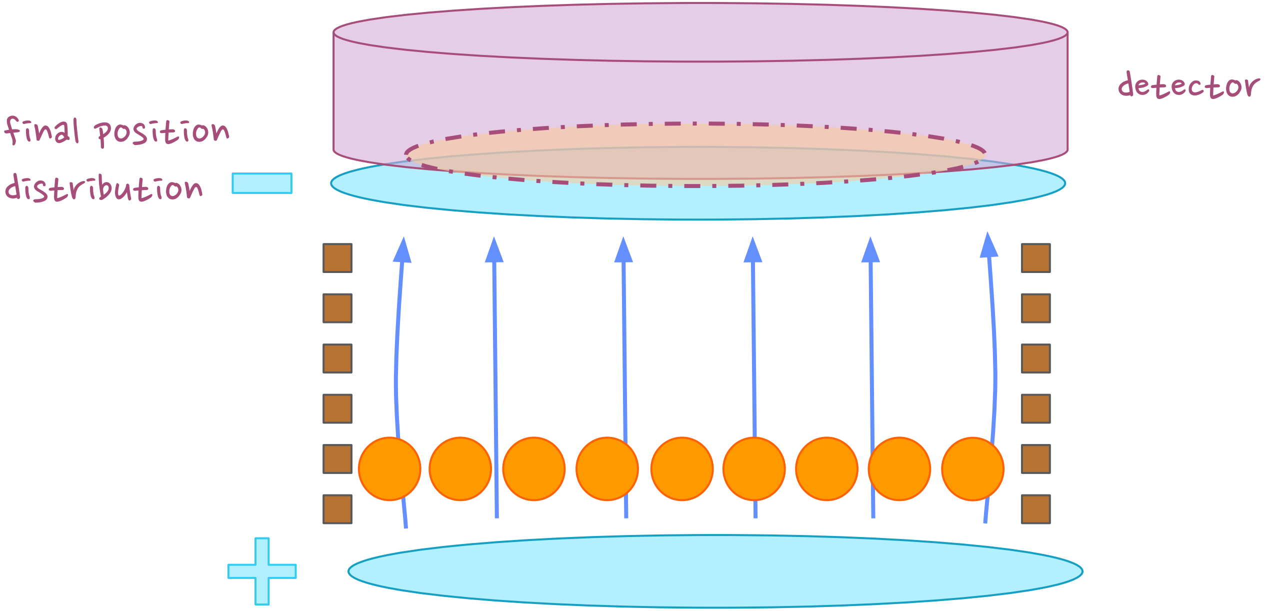
1. Can I learn where the electrons started?
2. Can I infer the electric field?
3. Can I quantify my uncertainties?

I know where my electrons ended up (sort of).

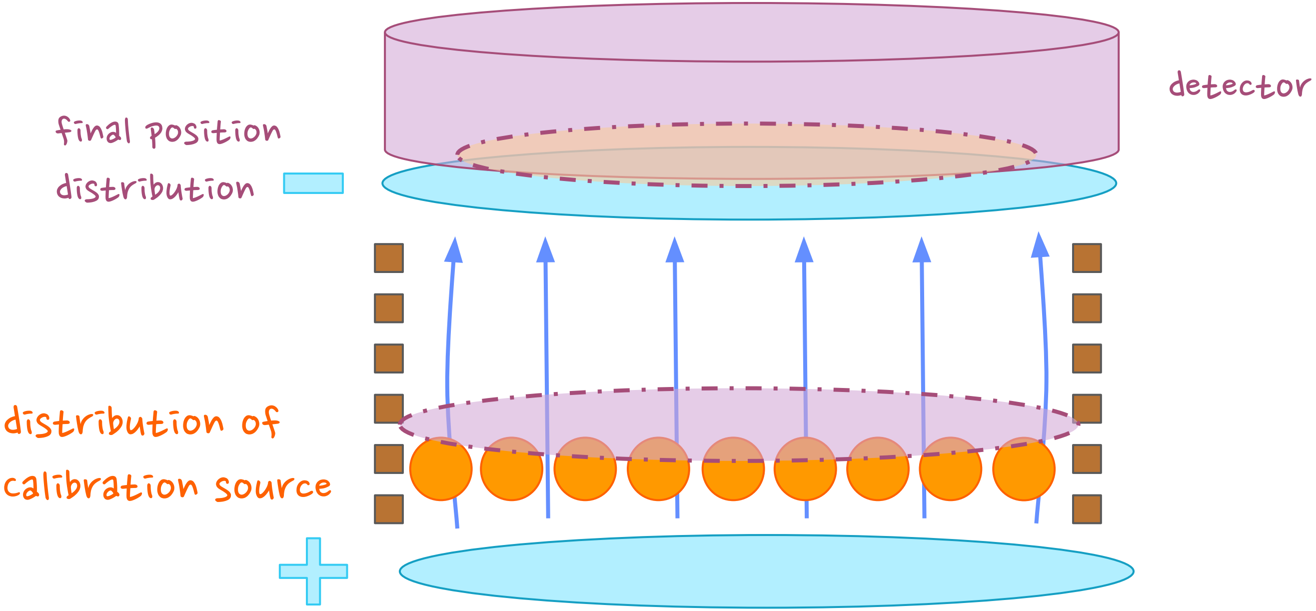


1. <u>Can I learn where the electrons started?</u>
2. Can I infer the electric field?
3. Can I quantify my uncertainties?

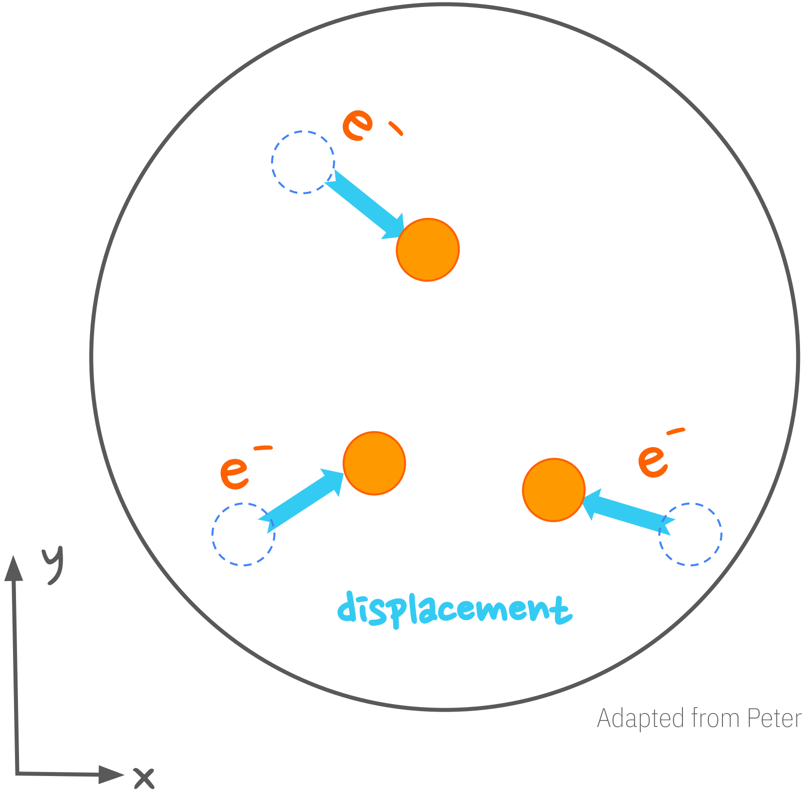

detector
−
+



final position
distribution
detector


detector
final position
distribution
distribution of
calibration source

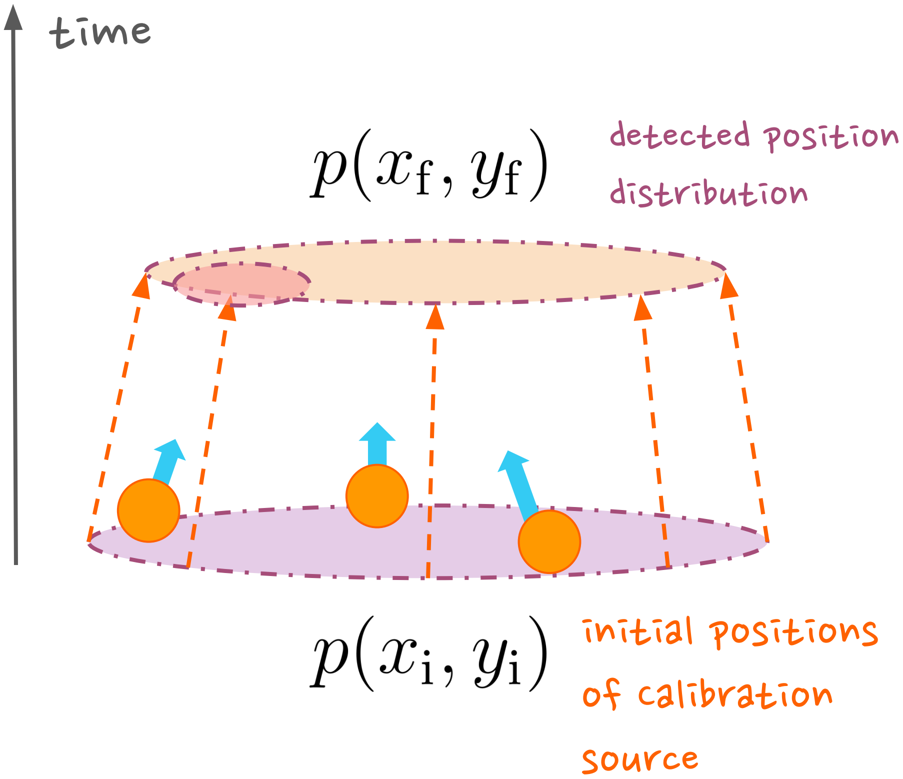
What we actually detect



Adapted from Peter Gaemers

We pop open the detector

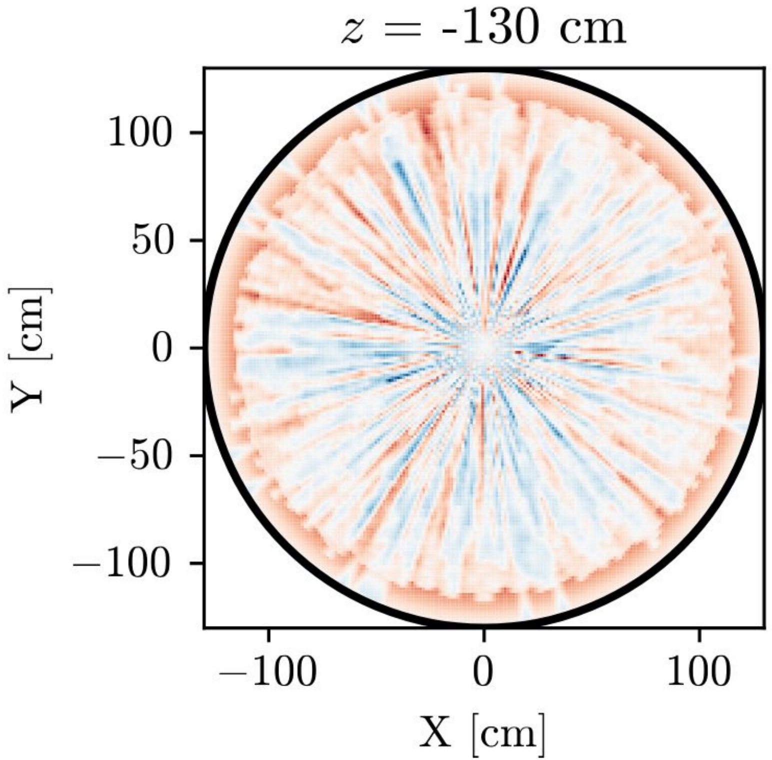
(this is unfeasible)

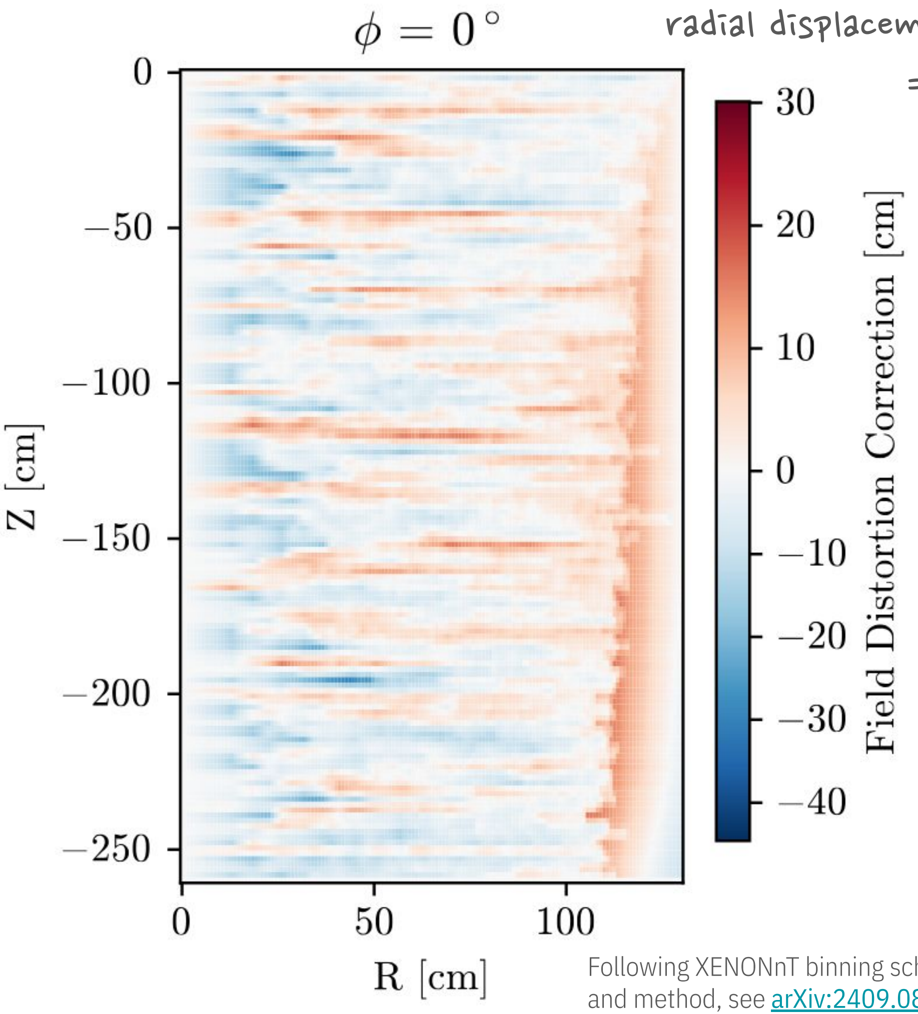


Field distortion correction (FDC) map that maps every given (x,y,z) to its radial correction

radial displacement

= FDC



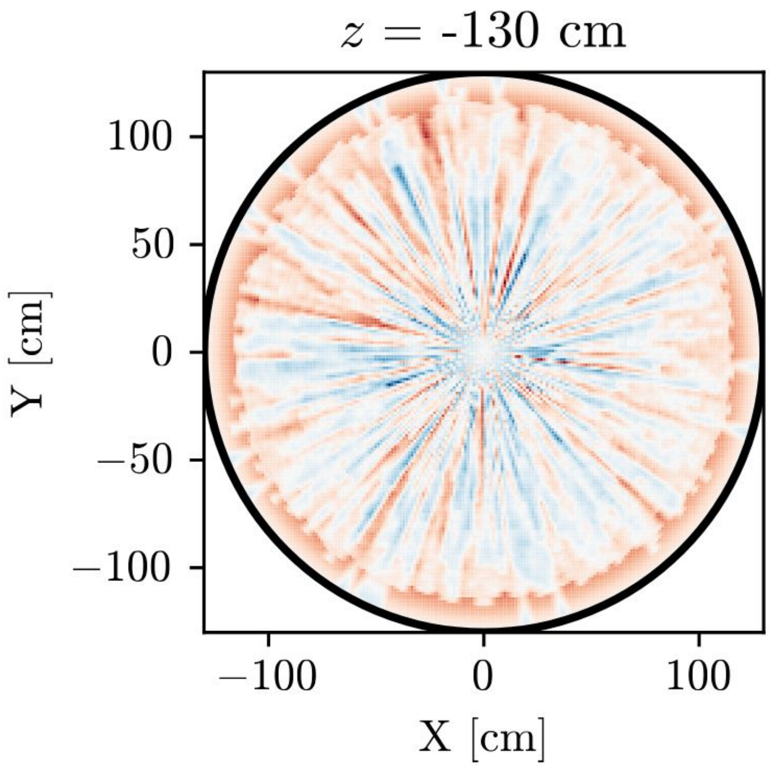


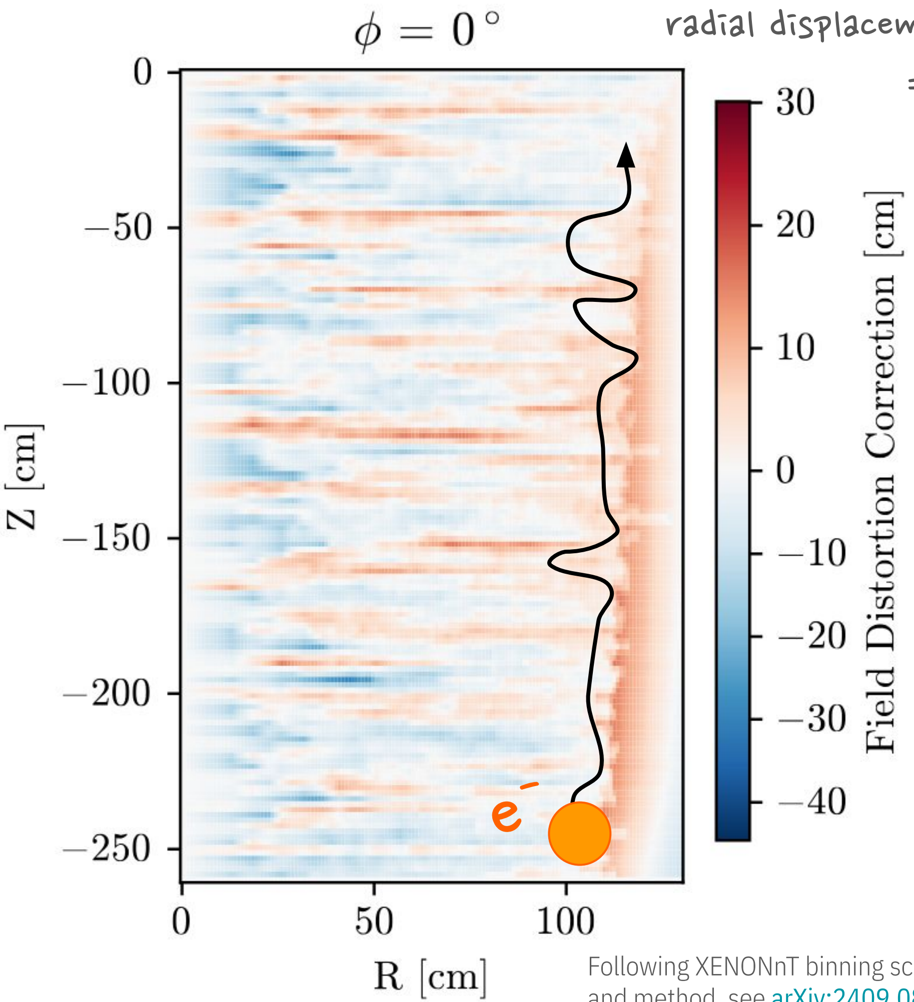
Following XENONnT binning scheme and method, see arXiv:2409.08778

This is terrible! 😱

radial displacement

= FDC





Following XENONnT binning scheme and method, see arXiv:2409.08778

This is terrible! 😱

(for the record, this is the electric field used for generating simulation data for this correction map)

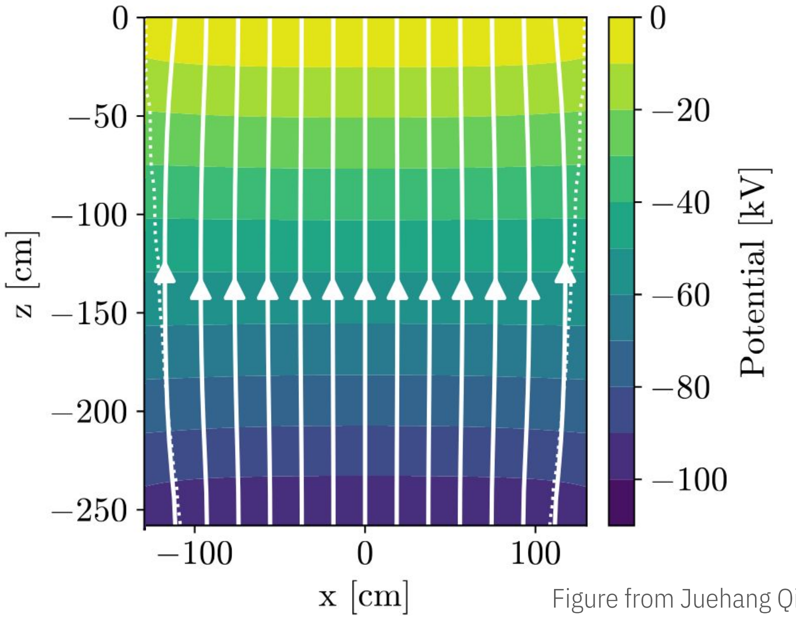
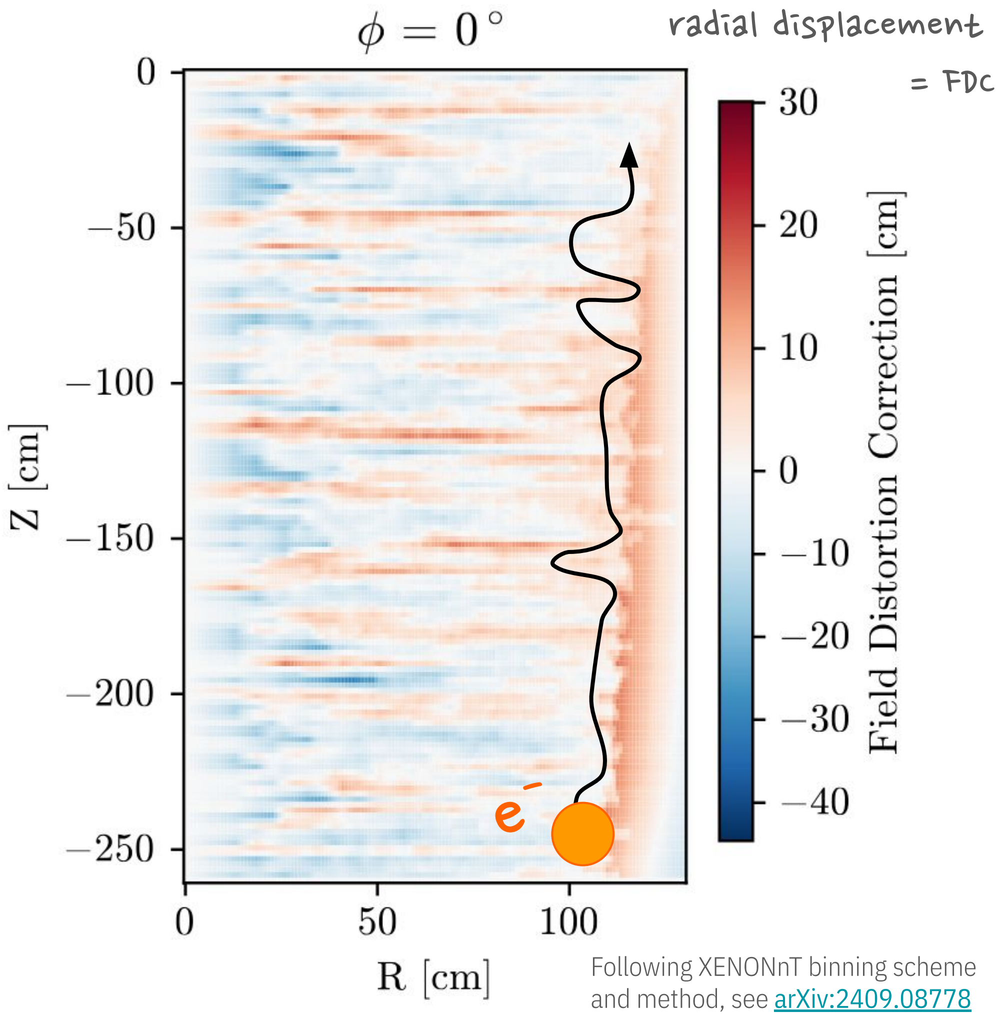


Figure from Juehang Qin



Following XENONnT binning scheme and method, see arXiv:2409.08778

This is terrible! 😱

(for the record, this is the electric field used for generating simulation data for this correction map)

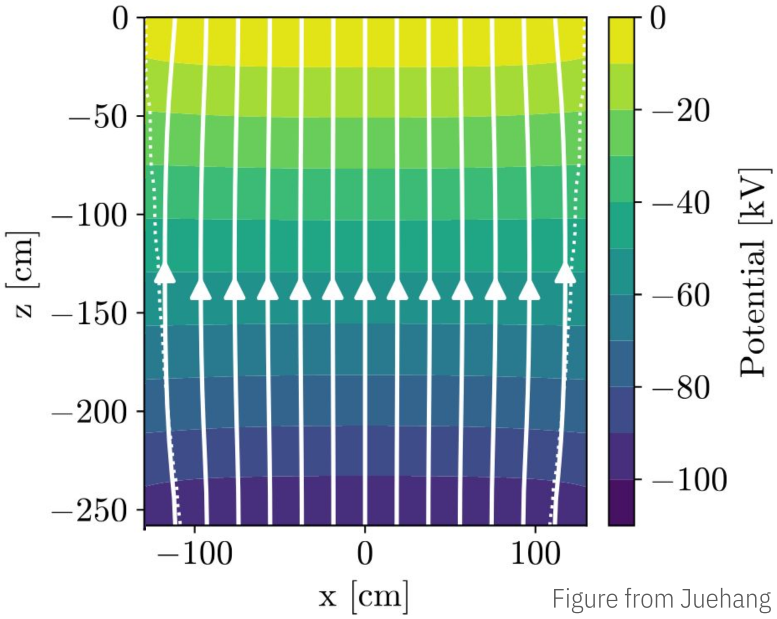
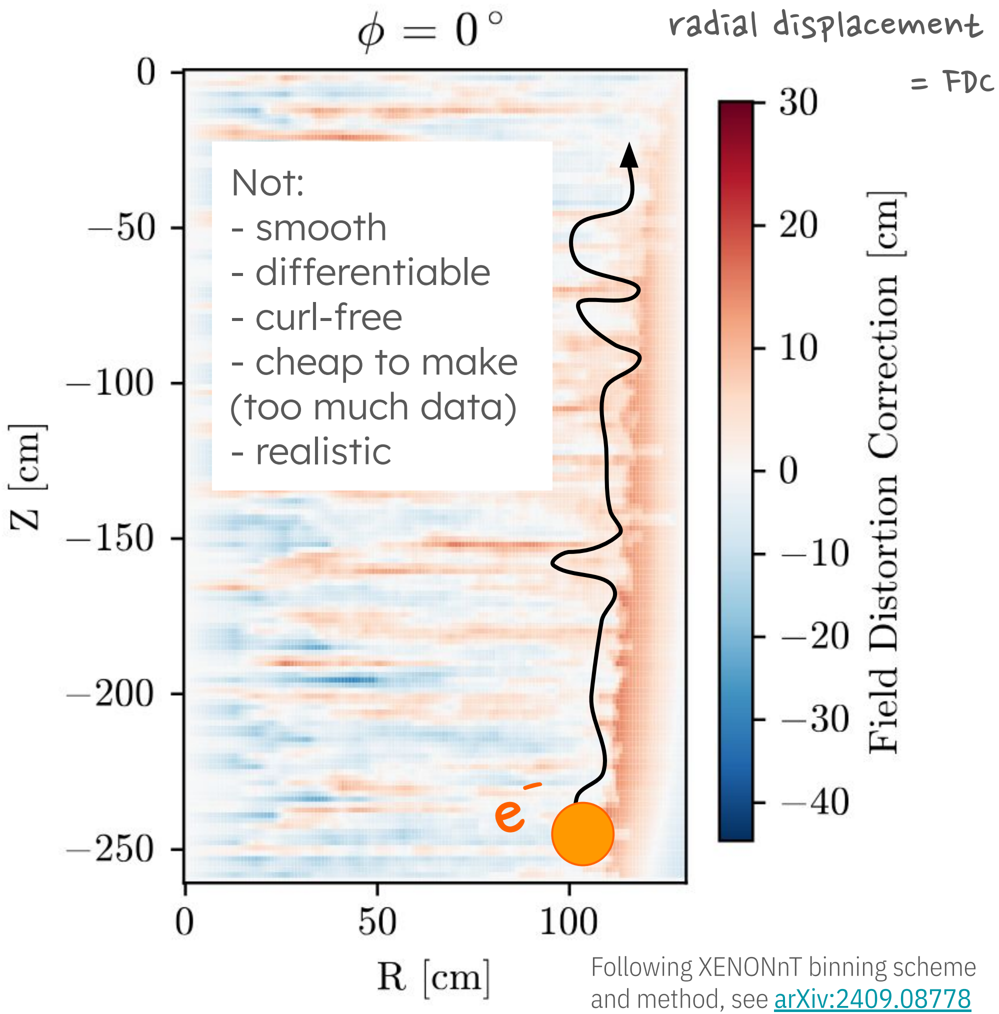


Figure from Juehang Qin



Following XENONnT binning scheme and method, see arXiv:2409.08778

This is terrible! 😱

(for the record, this is the electric field used for generating simulation data for this correction map)

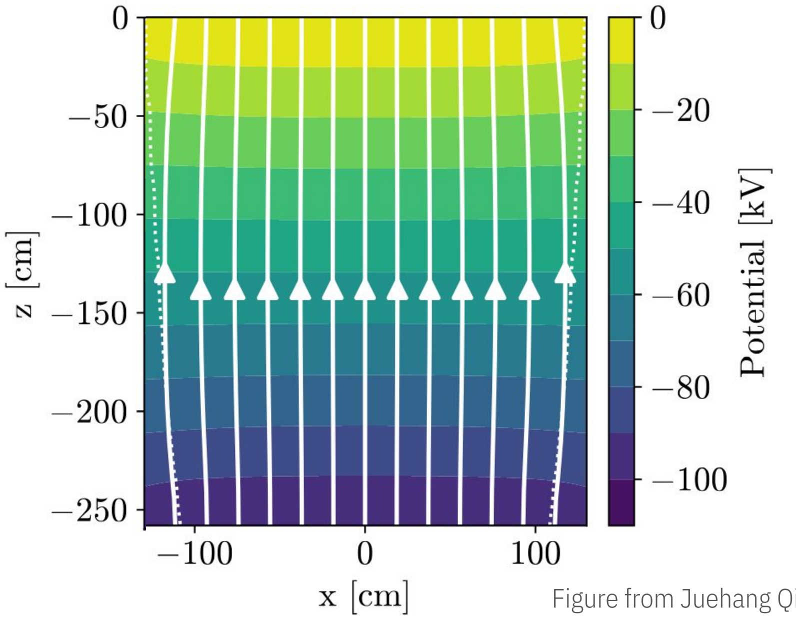
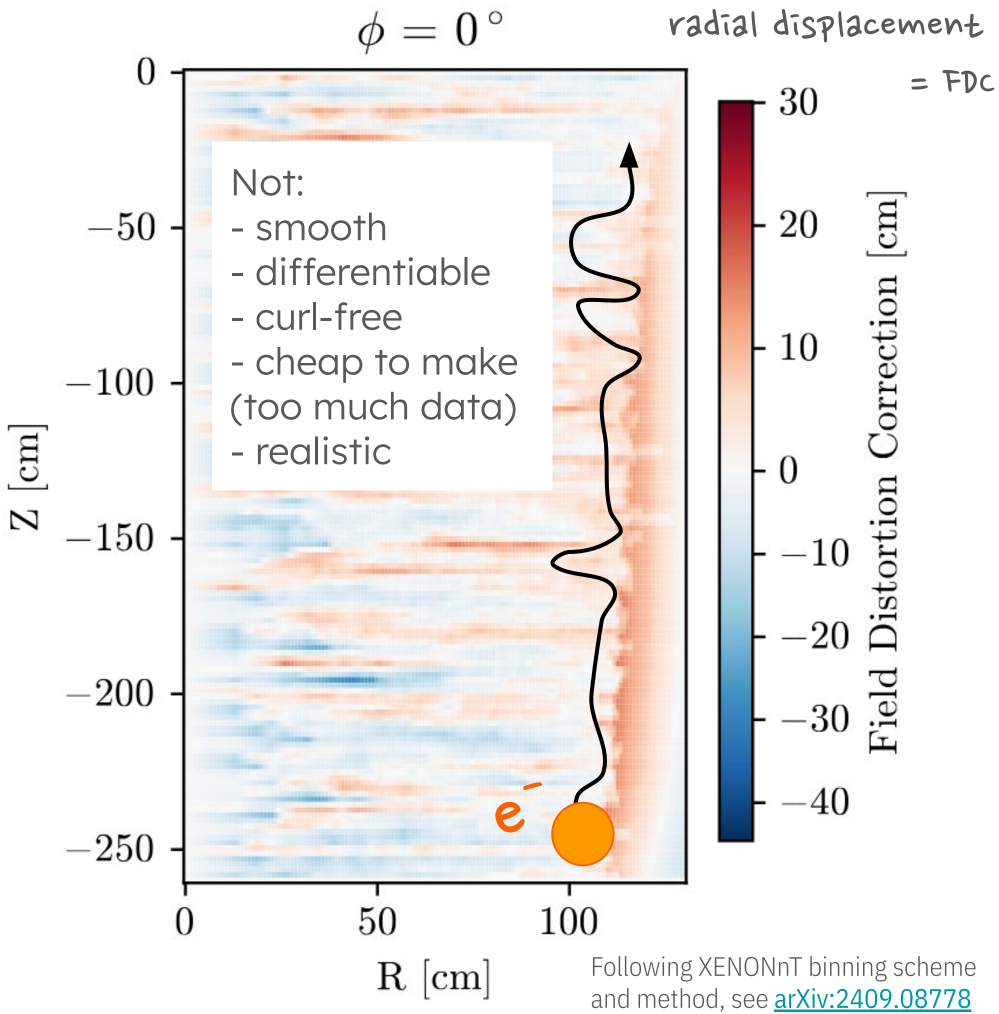
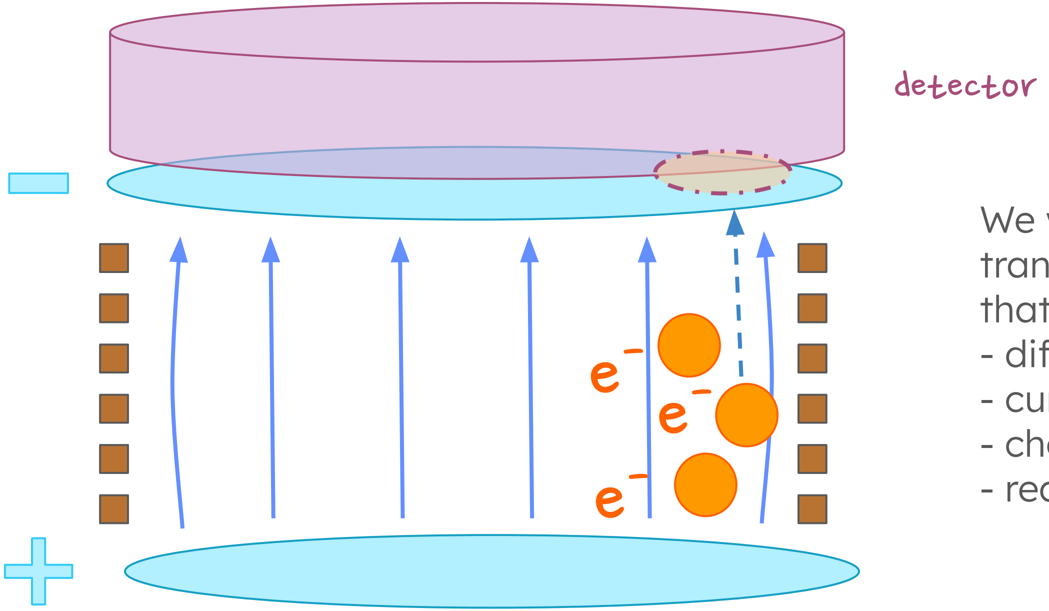


Figure from Juehang Qin



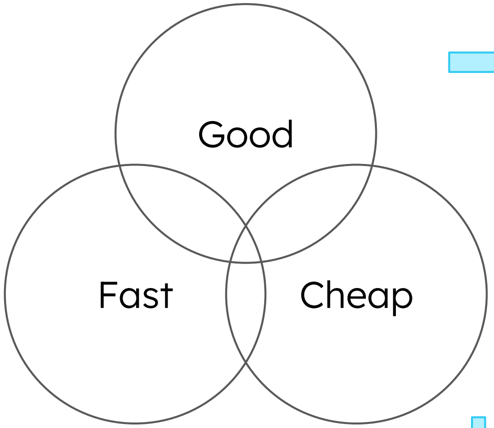
Following XENONnT binning scheme and method, see [arXiv:2409.08778](https://arxiv.org/abs/2409.08778)

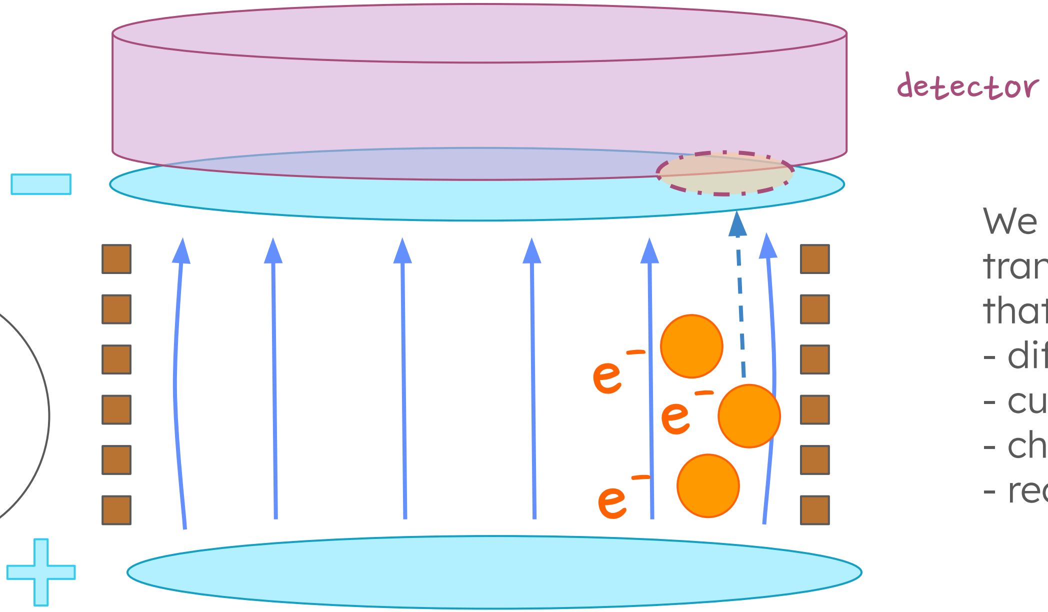
Can we do better?



We want a field transformation that is:
- differentiable
- curl-free
- cheaper to make
- realistic

1. Can I learn where the electrons started?
2. Can I infer the electric field?
3. Can I quantify my uncertainties?

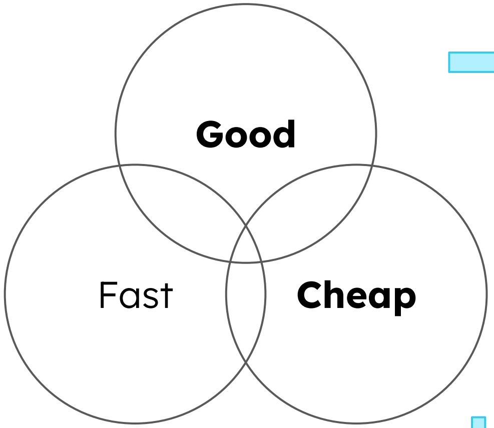


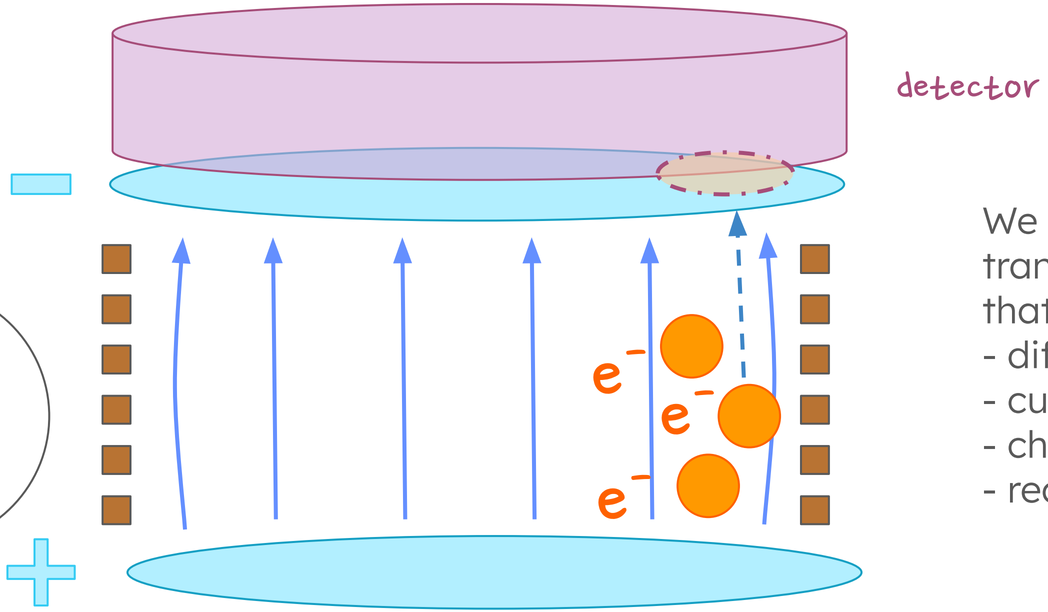


We want a field transformation that is:
- differentiable
- curl-free
- cheaper to make
- realistic

1. Can I learn where the electrons started?
2. Can I infer the electric field?
3. Can I quantify my uncertainties?

Can we search the space of physically possible electric fields to find the best transformation that matches our data?





We want a field transformation that is:
- differentiable
- curl-free
- cheaper to make
- realistic

1. Can I learn where the electrons started?
2. Can I infer the electric field?
3. Can I quantify my uncertainties?

Can we search the space of physically possible electric fields to find the best transformation that matches our data?

continuous

normalizing flow

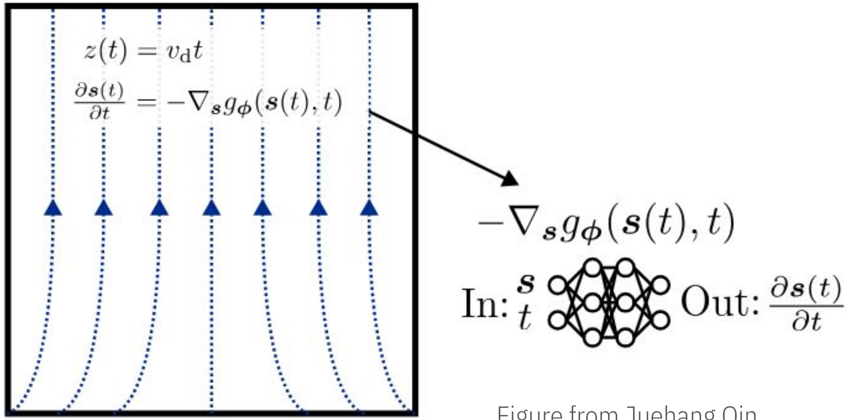
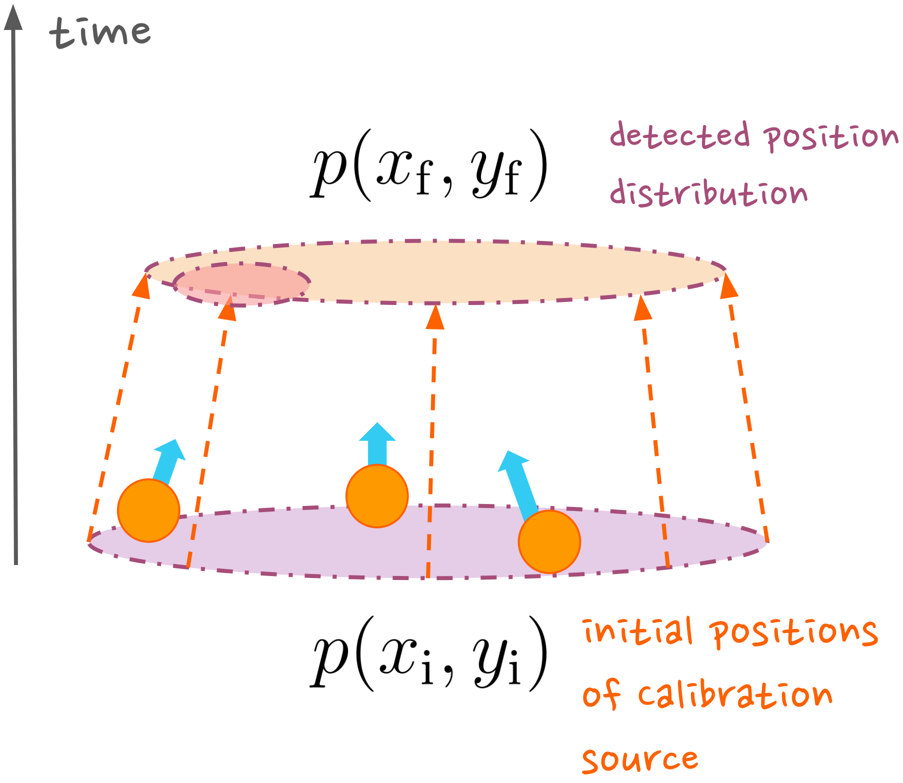


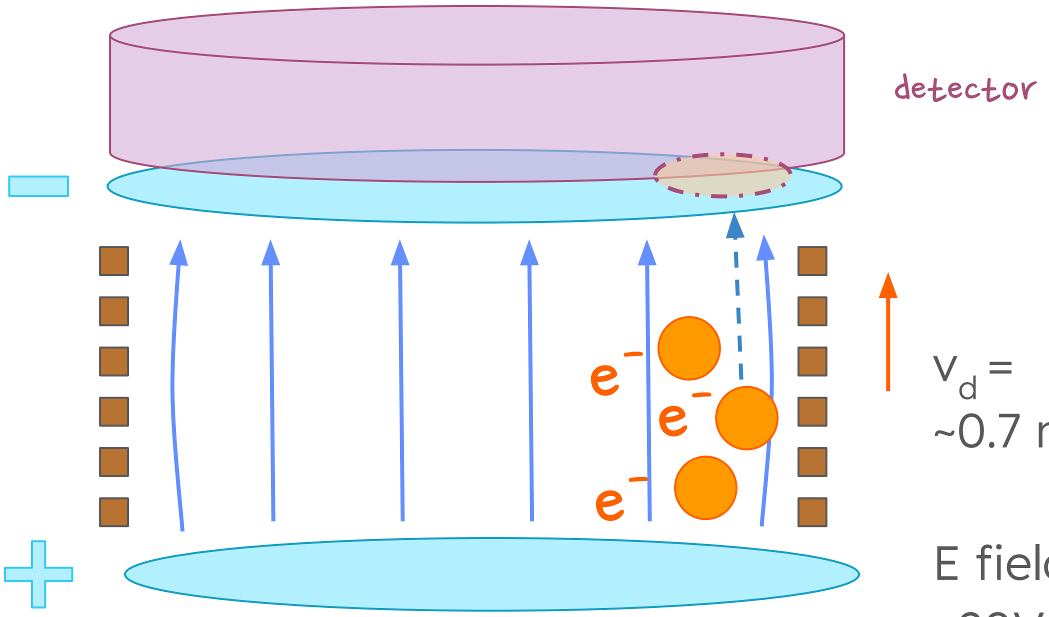
Figure from Juehang Qin



Neural ODE: arXiv:1806.07366

Normalizing Flows for Inference: arXiv:1912.02762

Let's build a simulator! What is an example of a worst case field?



v_d =
~0.7 mm/µs

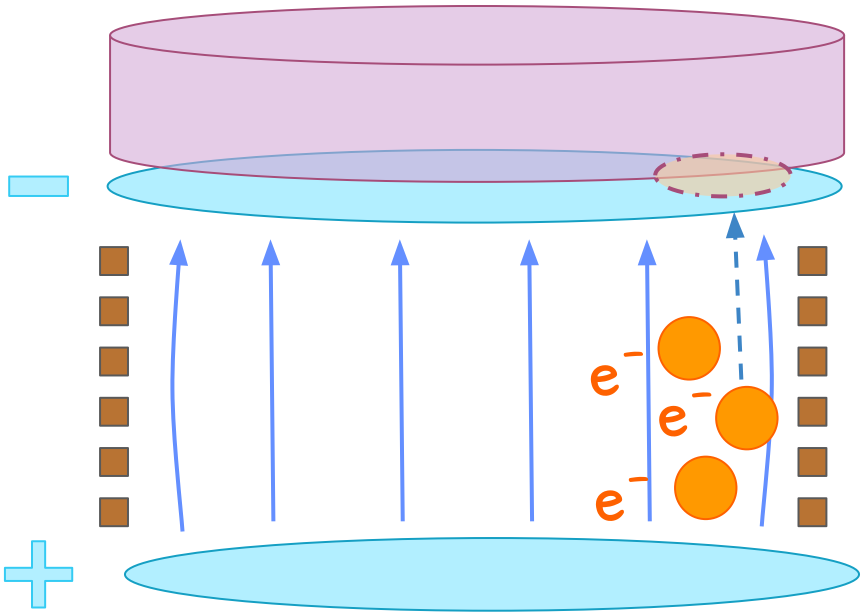
E field is
~20V/cm

Slow E-field: e- take 2.2 ms to travel the whole detector in XENONnT

Following XENONnT electron drift velocity and field, see arXiv:2409.08778

Let’s build a simulator! What is an example of a worst case field?

E-field is fairly vertically straight, let’s assume z correction is minimal:

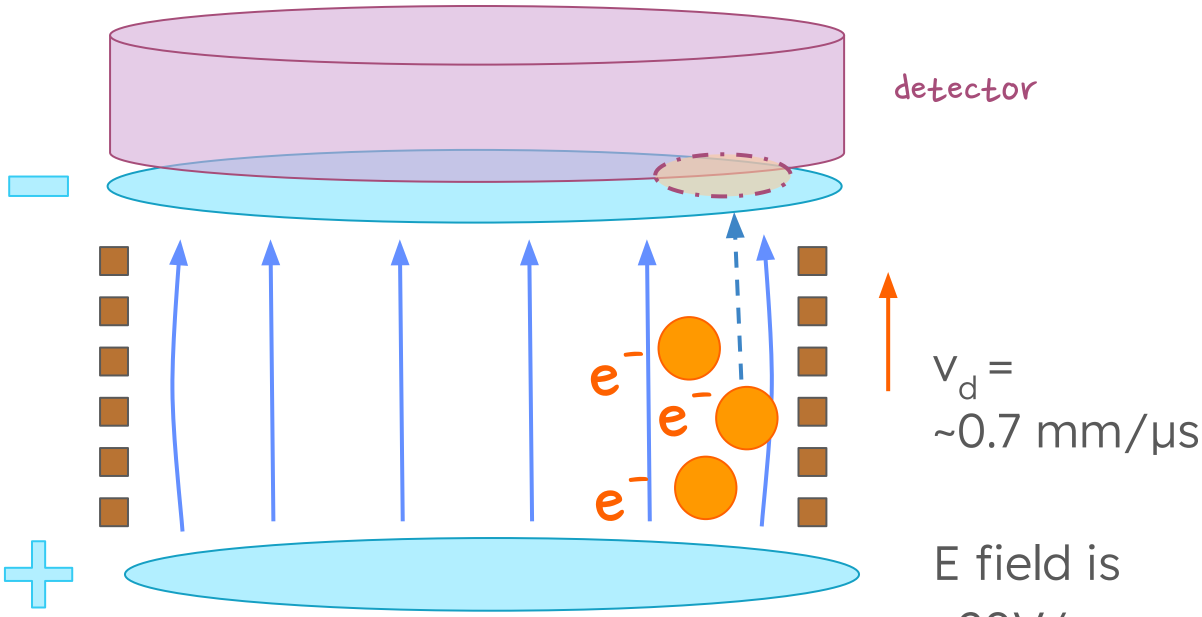


$$
\begin{aligned}
\frac{\partial \boldsymbol{s}(t)}{\partial t} &= -\mu_{\mathrm{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), z) \\
&= -\mu_{\mathrm{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), z_0 + v_{\mathrm{d}} t)
\end{aligned}
$$

Slow E-field: e- take 2.2 ms to travel the whole detector in XENONnT

Following XENONnT electron drift velocity and field, see arXiv:2409.08778

Let's build a simulator! What is an example of a worst case field?



We can solve this differential equation!

$$\frac{\partial \boldsymbol{s}(t)}{\partial t} = -\mu_{\mathrm{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), z)$$
$$= -\mu_{\mathrm{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), z_0 + v_{\mathrm{d}} t)$$

Slow E-field: e- take 2.2 ms to travel the whole detector in XENONnT

Following XENONnT electron drift velocity and field, see arXiv:2409.08778

Let’s be more clever:

$$\frac{\partial \boldsymbol{s}(t)}{\partial t} = -\mu_{\mathrm{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), z)$$
$$= -\mu_{\mathrm{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), z_0 + v_{\mathrm{d}} t)$$

We want a field transformation that is:
- **differentiable**
- curl-free
- cheaper to make
- realistic

Let's be more clever:

$$\frac{\partial \boldsymbol{s}(t)}{\partial t} = -\mu_{\text{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), z)$$
$$= -\mu_{\text{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), z_0 + v_{\text{d}} t)$$

Let's learn the scalar potential instead!

$$\frac{\partial \boldsymbol{s}(t)}{\partial t} = -\nabla_{\boldsymbol{s}} g_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t)$$

We want a field transformation that is:
- **differentiable**
- **curl-free**
- cheaper to make
- realistic

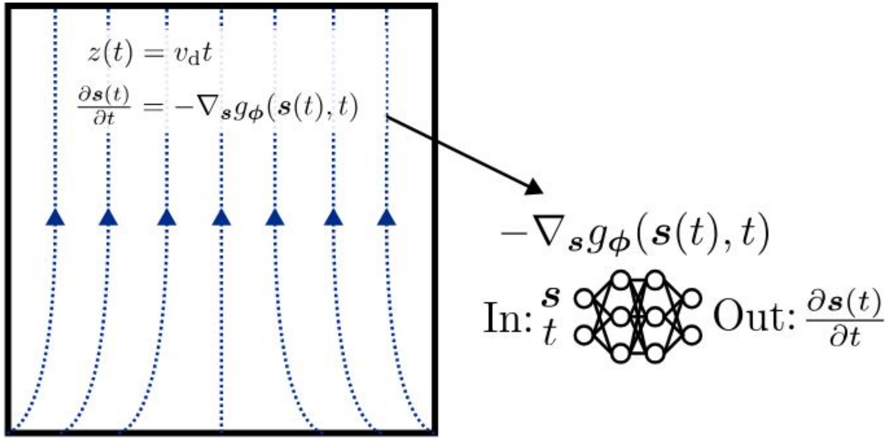
Let's be more clever:

$$\frac{\partial \boldsymbol{s}(t)}{\partial t} = -\mu_{\text{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), z) = -\mu_{\text{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), z_0 + v_{\text{d}} t)$$

Let's learn the scalar potential instead!

$$\frac{\partial \boldsymbol{s}(t)}{\partial t} = -\nabla_{\boldsymbol{s}} g_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t)$$

We want a field transformation that is:
- **differentiable**
- **curl-free**
- cheaper to make
- realistic



Let's be more clever:

$$\frac{\partial \boldsymbol{s}(t)}{\partial t} = -\mu_{\mathrm{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), z)$$
$$= -\mu_{\mathrm{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), z_0 + v_{\mathrm{d}} t)$$

We want a field transformation that is:

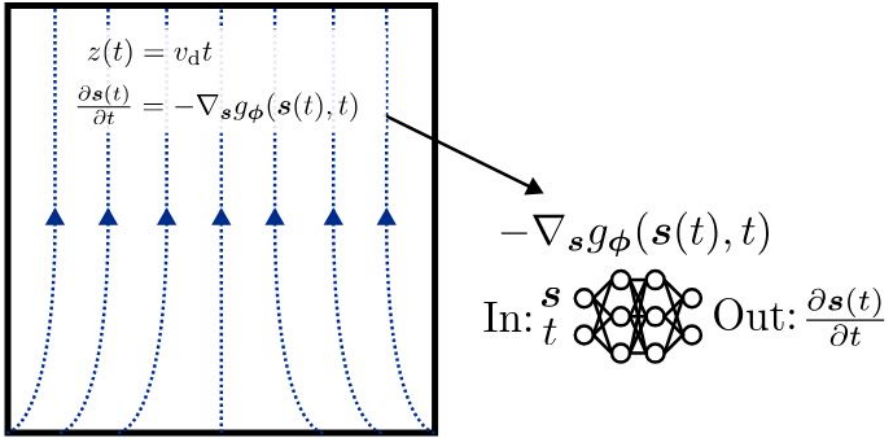
- **differentiable**
- **curl-free**
- cheaper to make
- realistic

Let's learn the scalar potential instead!

$$\frac{\partial \boldsymbol{s}(t)}{\partial t} = -\nabla_{\boldsymbol{s}} g_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t)$$

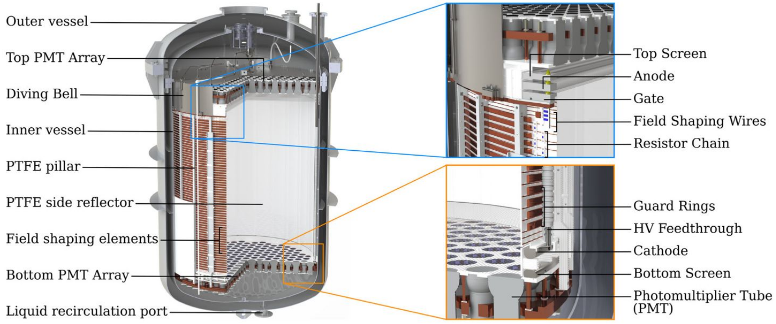
To move through our model, we solve the ODE:

$$\boldsymbol{s}(t) = \boldsymbol{s}(t_0) + \int_{t_0}^{t} g_{\phi}(\boldsymbol{s}(\tau), \tau) d\tau$$

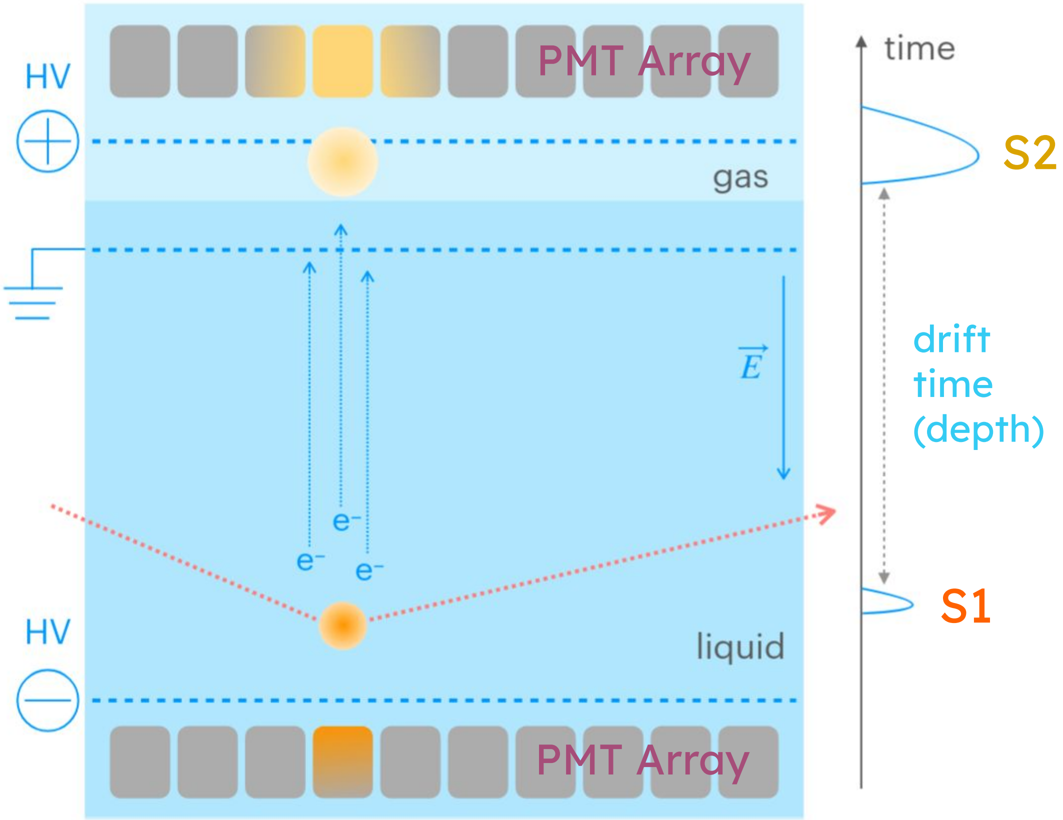


We have an idea!
Let's apply this to a realistic simulator of a xenon TPC.

The ionization electron starts from the first scintillation signal (S1) and ends at the second ionization (S2) signal.



(Left) arXiv:2402.10446

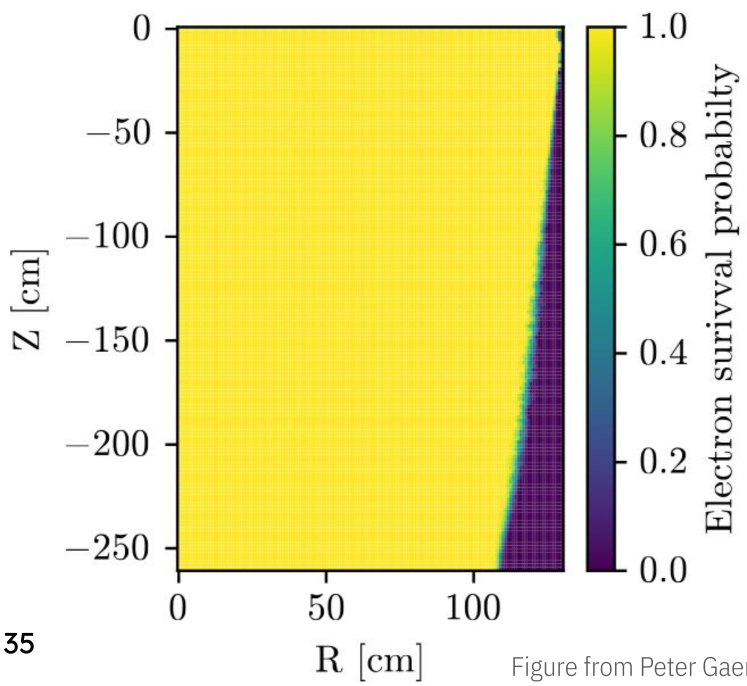


The LXeTPC has a liquid layer and a gaseous layer of xenon, electric fields, and PMT arrays above and below the detector.

(Right) adapted from arXiv:2404.19524

We pop in a calibration source that diffuses uniformly and simulate an expected electron survival probability map.

What we expect
(initial position dist):



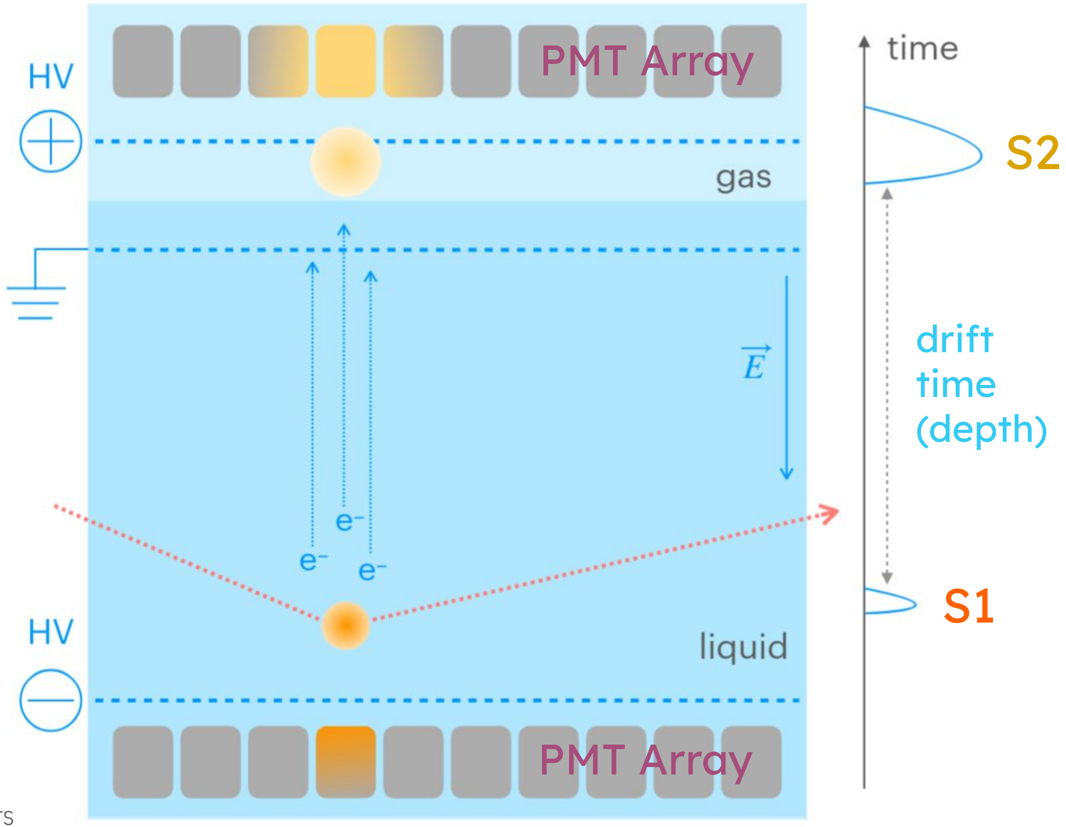
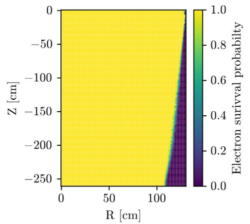
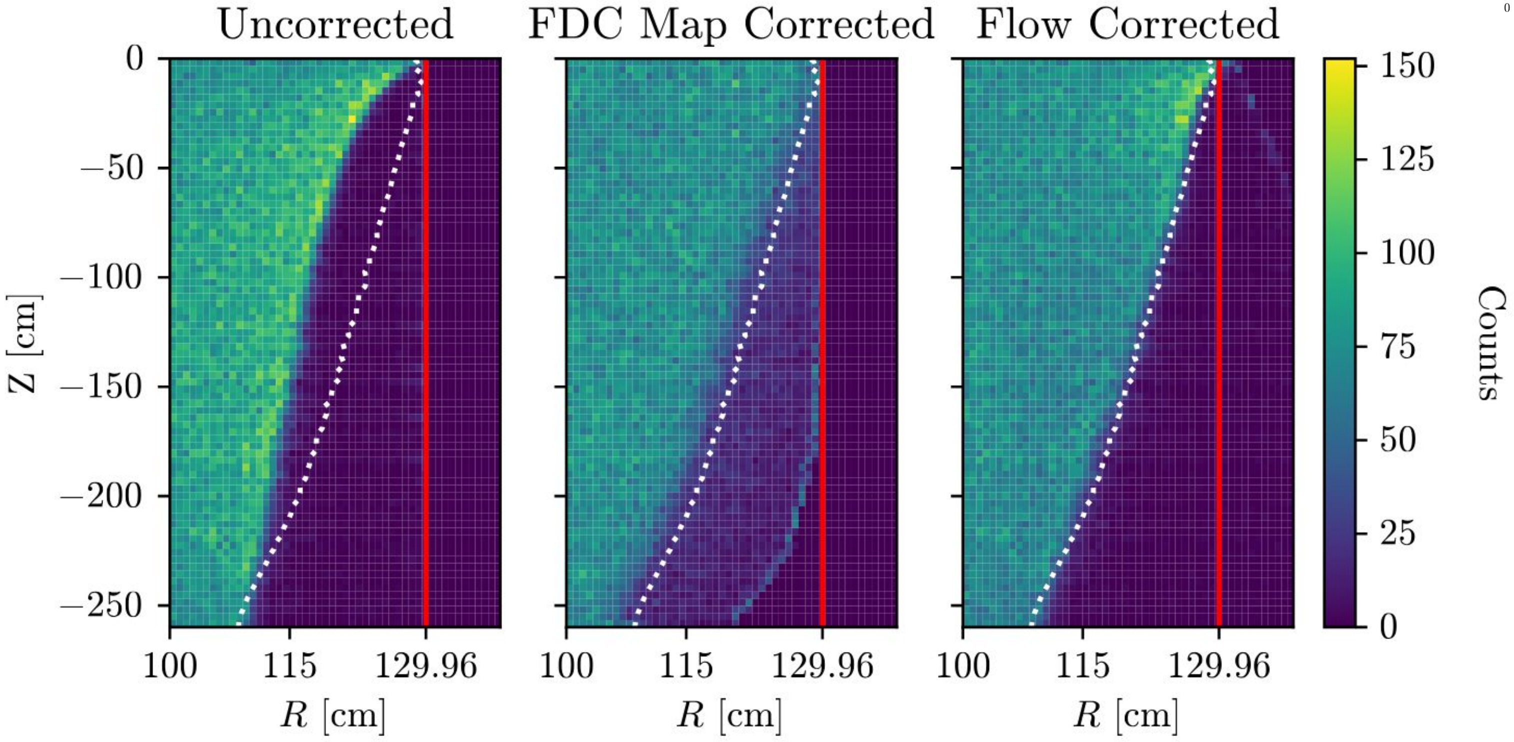


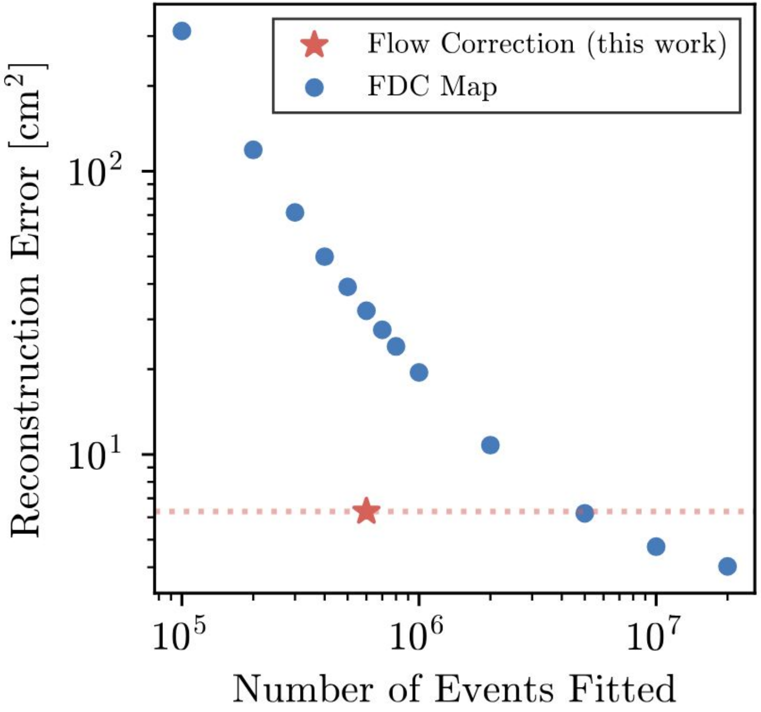
Figure from Peter Gaemers

We recover the expected position distribution!



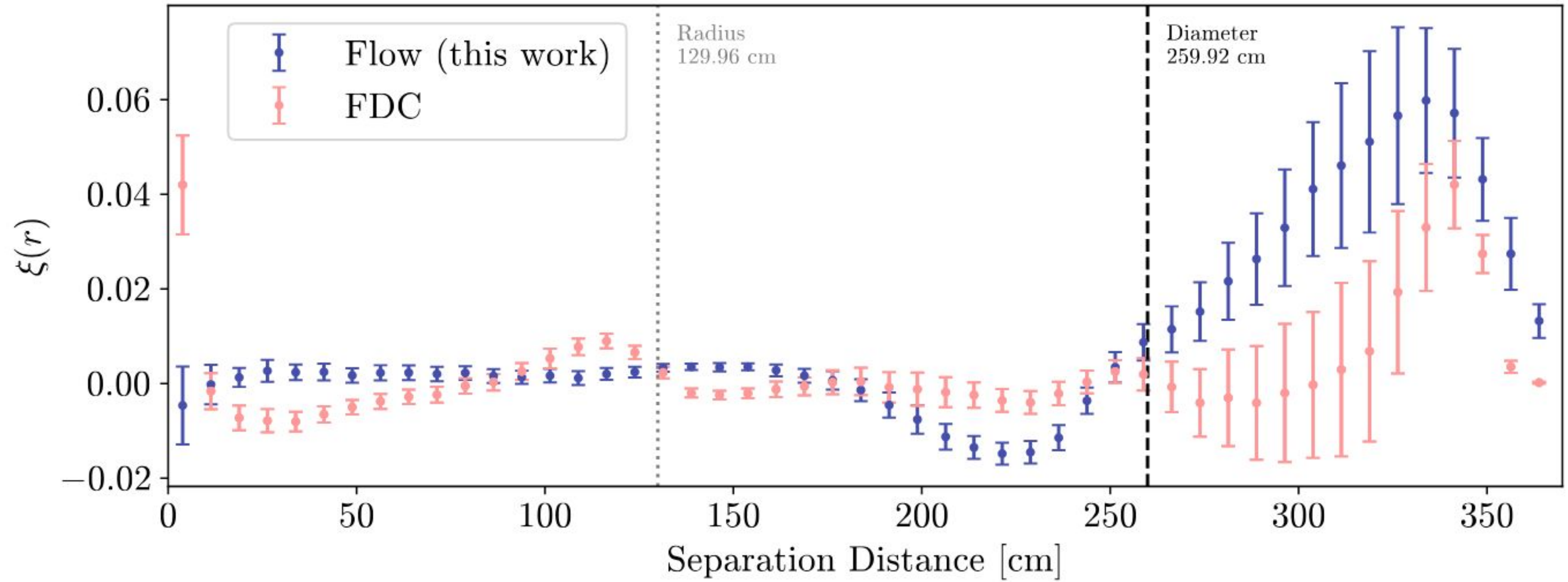


We need an order of magnitude less data with the flow!



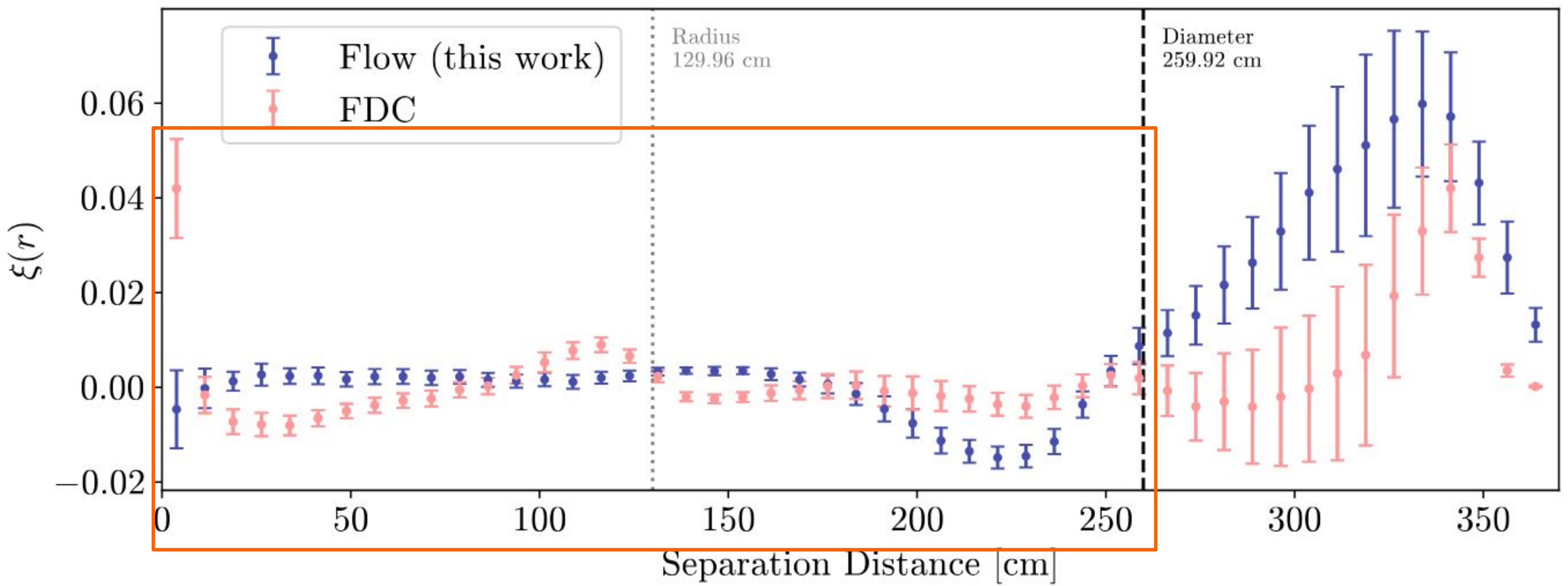
We want a field transformation that is:
- **differentiable**
- **curl-free**
- **cheaper to make**
- realistic

And we preserve local clustering patterns better than the FDC.



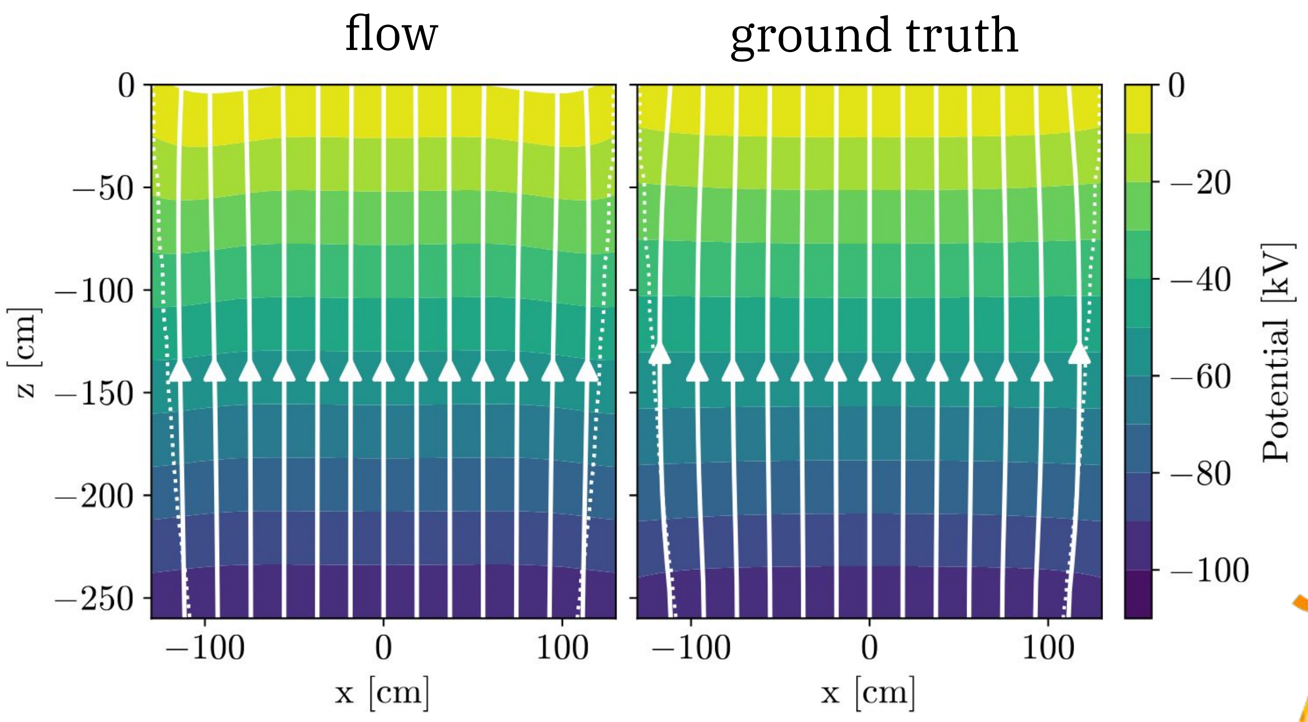
*normalized by subtracting ground truth 2 point correlation function

And we preserve local clustering patterns better than the FDC.



*normalized by subtracting ground truth 2 point correlation function

The ultimate question…do we get back the electric field? 🥁



Yes! We can infer the electric field using machine learning!

Can we use this to quantify and propagate uncertainties?

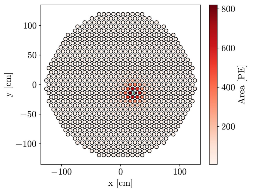
(hit_pattern, t_{S1}, t_{S2}, ...)

S2 position reconstruction

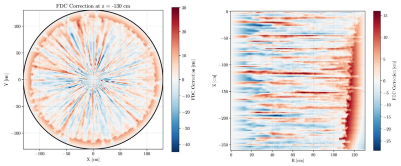
(x_{S2}, y_{S2}, drift_time)

we make corrections :)

(x_{int}, y_{int}, z_{int})



Maximum likelihood fitter (LZ) or train a dense NN (XENONnT)
Input: hit pattern
Output: (x_{S2},y_{S2})



Data-Driven Field Distortion Correction Map (LUX + XENON)
Uniform monoenergetic calibration source
→ bin the data
→ find radial correction per bin
→ interpolate the map

Yes it is possible to build a fully probabilistic framework for position reconstruction!

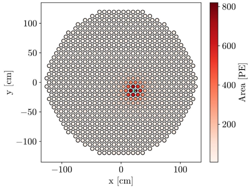
(hit_pattern, t_{S1}, t_{S2}, ...)

S2 position reconstruction

(x_{S2}, y_{S2}, drift_time)

we make corrections :)

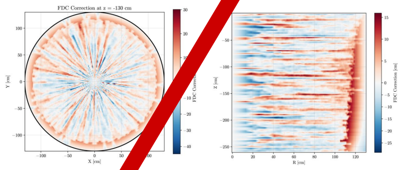
(x_{int}, y_{int}, z_{int})



Maximum likelihood fitter (LZ) or train a dense NN (XENONnT)

Input: hit pattern

Output: (x_{S2}, y_{S2})



Data-Driven Field Distortion Correction Map (LUX + XENON)

Uniform monoenergetic calibration source

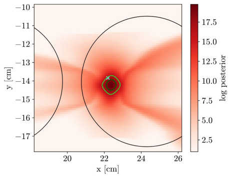
→ bin the data

→ find radial correction per bin

→ interpolate the map

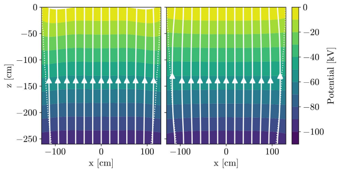
Conditional normalizing flows for probabilistic position reconstruction

Based on work from Sebastian Vetter and Juehang Qin (paper and repo forthcoming)



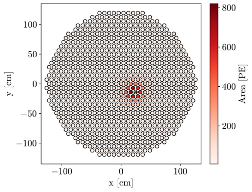
Continuous normalizing flows for determining electric fields in TPCs (this work!)

Paper and repo forthcoming, stay tuned!



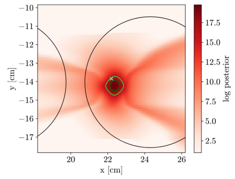
Future work is in making our data pipeline probabilistic to have proper uncertainty quantification and propagation.

`(hit_pattern,` t_{S1}, t_{S2}`,...)`

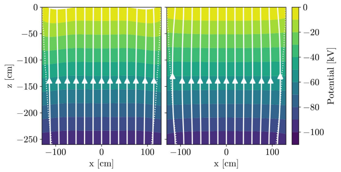


S2 probabilistic position

`(` x_{S2}, y_{S2}`,drift_time)`

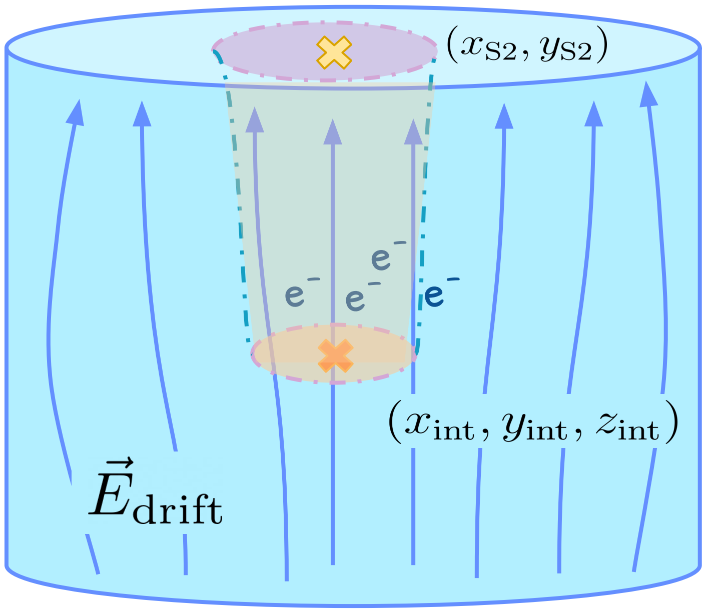


differentiable corrections



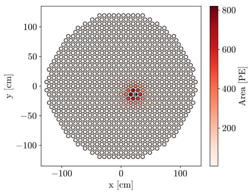
`(` x_{int}, y_{int}, z_{int}`)`

get initial position ± uncertainty



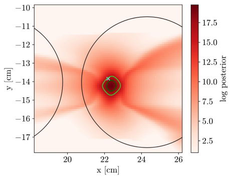
Future work is in making our data pipeline probabilistic to have proper uncertainty quantification and propagation.

$(\mathtt{hit\_pattern}, t_{S1}, t_{S2}, \ldots)$

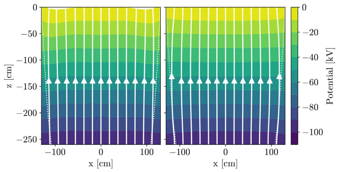


S2 probabilistic position

$(x_{S2}, y_{S2}, \mathtt{drift\_time})$



differentiable corrections



$(x_{int}, y_{int}, z_{int})$

get initial position ± uncertainty

We want a field transformation that is:
- **differentiable**
- **curl-free**
- **cheaper to make**
- **realistic**

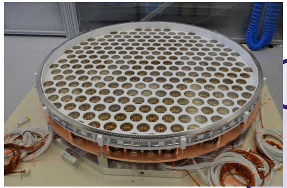
Got a TPC? Collaborate with us!

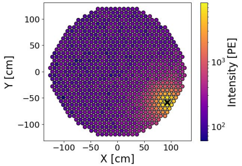
Ivy Li (il11@rice.edu)
Peter Gaemers (pgaemers@stanford.edu)
Juehang Qin (qinjuehang@rice.edu)
Naija Bruckner (naija.bruckner@rice.edu)

arxiv preprint on the way, stay tuned

Backup

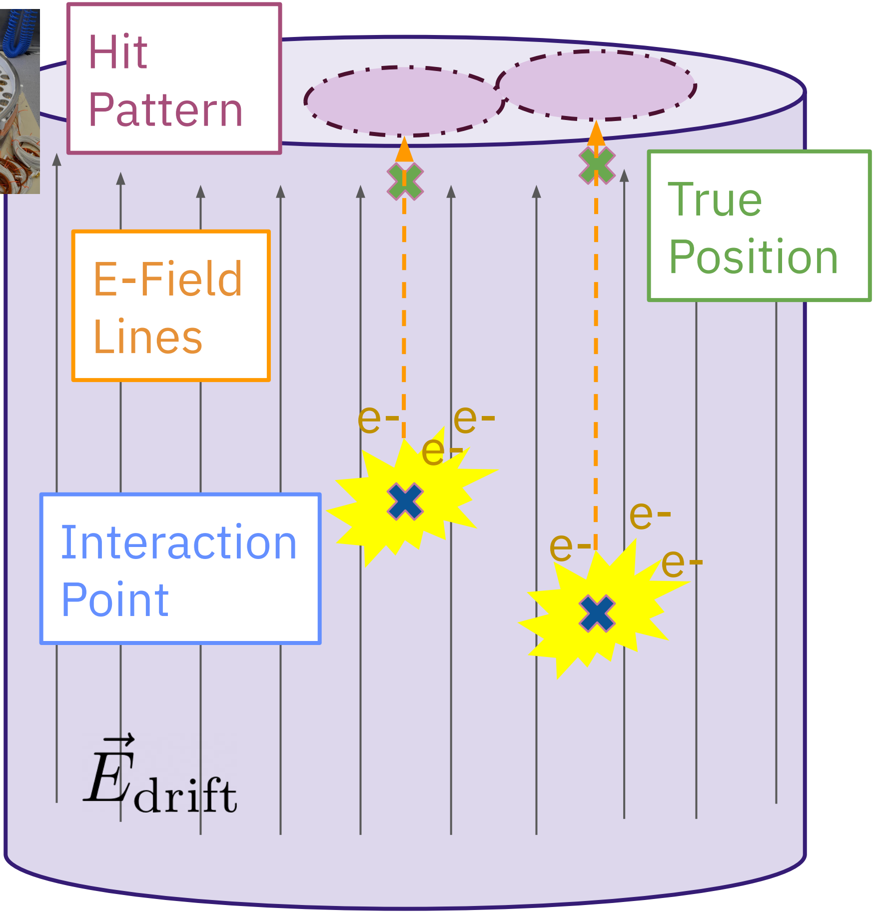
In particle detector experiments, we have known (observed data) and unknown parameters.





- Known:
    - light measured by photosensors, termed hit patterns

- Unknown:
    - true position at top of detector (x,y)
    - electric field lines
    - true interaction position (x_i,y_i)

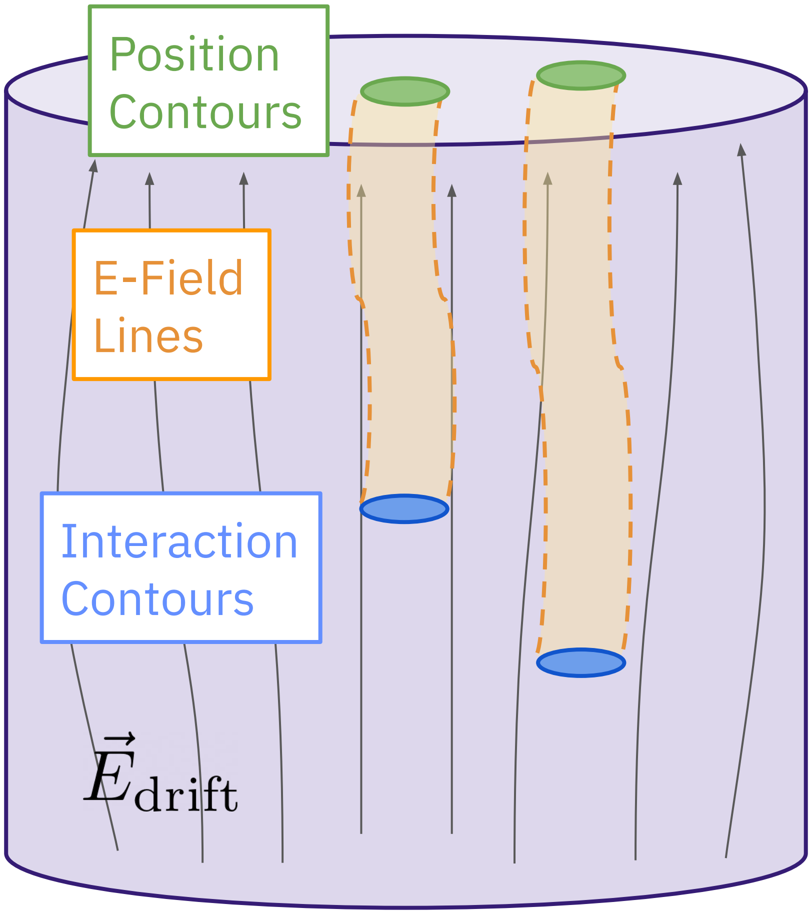


Future XLZD with perfect electric field :)

Moreover, there are uncertainties for each measurement, e.g. position uncertainty contour.

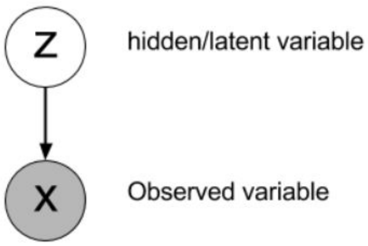
- Known:
    - light measured by photosensors, termed hit patterns

- Unknown:
    - true position at top of detector (x,y)
    - electric field lines
    - true interaction position (x_i,y_i)



Future XLZD with imperfect electric field :(

How do we infer these unknown parameters?



likelihood prior

$$p(Z|X) = \frac{p(X|Z)p(Z)}{p(X)}$$

posterior

normalization term

$$p(X) = \int_{z \in Z} p(X|z)p(z)\mathrm{d}z$$

Simulations-based inference

Infer parameters from data by using a simulator: give a simulator some parameters and generate data from it.

arXiv:1911.01429

Variational inference

Using optimization techniques (e.g. machine learning models) to approximate complex probability distributions (e.g. posterior probability density).

arXiv:2108.13083

In a perfect world, we would immediately recover the interaction position without corrections.

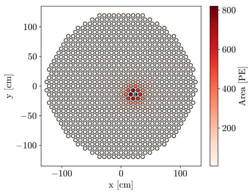
(hit_pattern, t_{S1}, t_{S2}, ...)

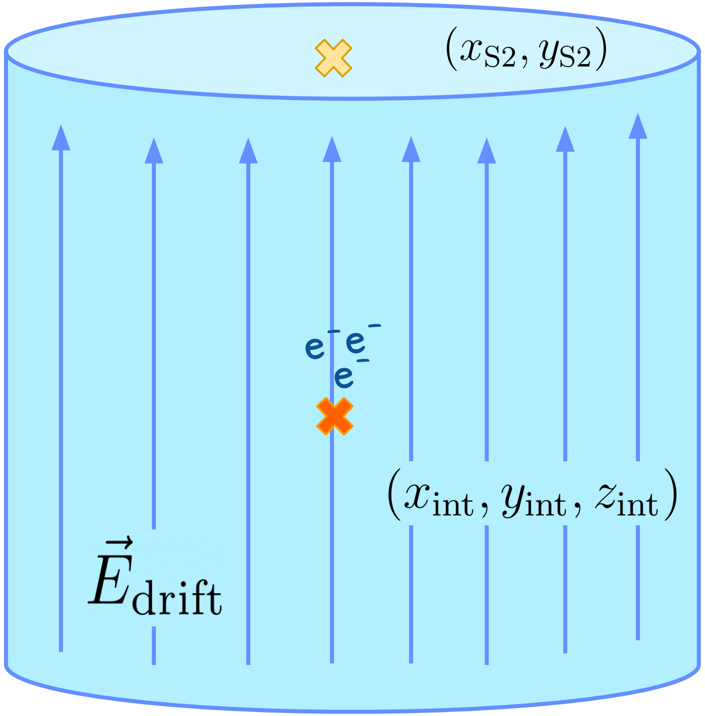
S2 position reconstruction

(x_{S2}, y_{S2}, drift_time)

no corrections :)

(x_{int}, y_{int}, z_{int})



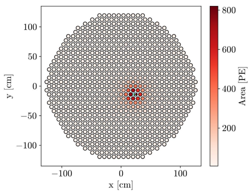


Future XLZD experiment! 😊

But we are not so perfect.

(hit_pattern, t_{S1}, t_{S2}, ...)

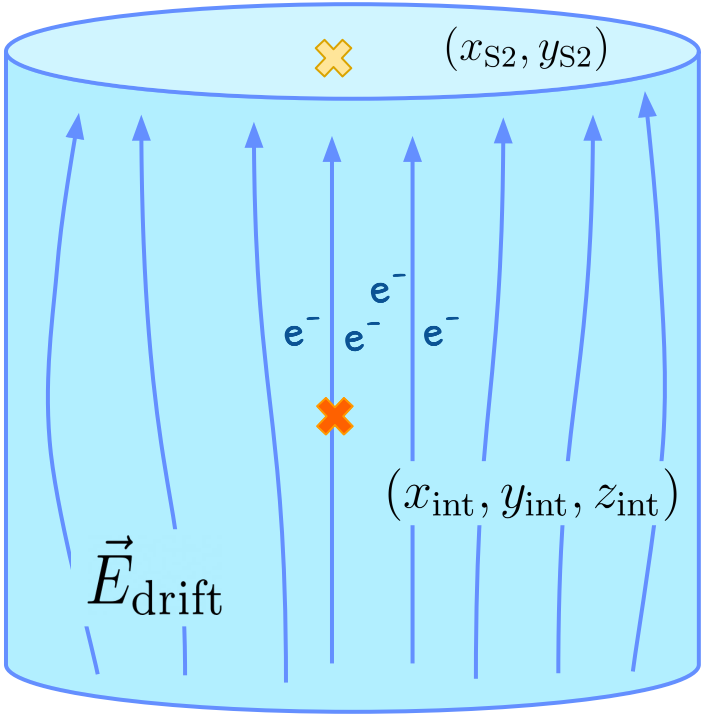
S2 position reconstruction



(x_{S2}, y_{S2}, drift_time)

we need corrections :(

(x_{int}, y_{int}, z_{int})



Future XLZD experiment? 🥲

Charge build up on the PTFE walls and the geometry of the detector result in an imperfect electric field.

(hit_pattern, t_{S1}, t_{S2}, ...)

S2 position reconstruction

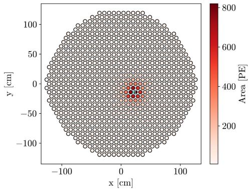
↓

(x_{S2}, y_{S2}, drift_time)

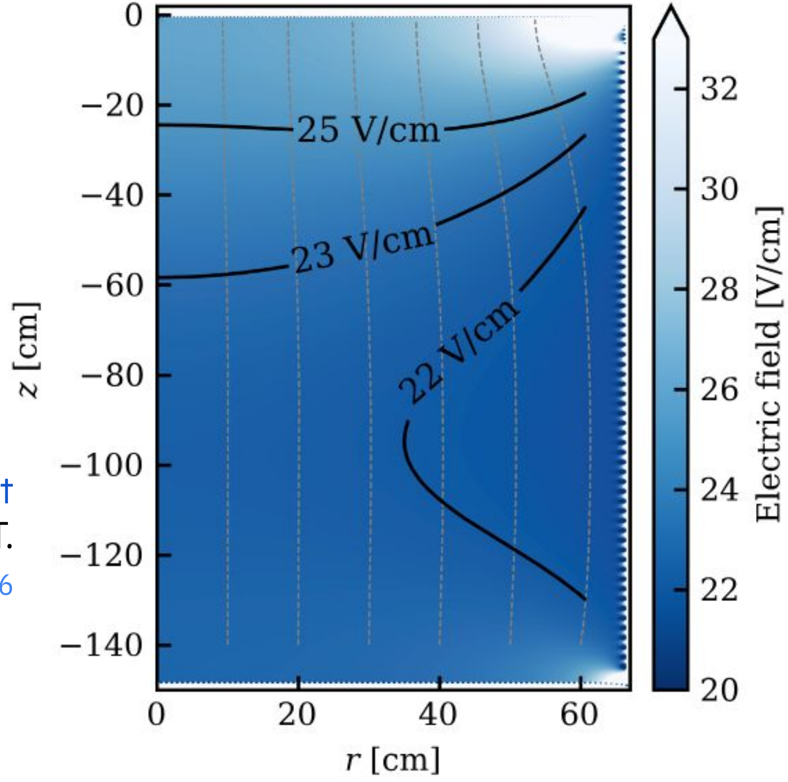
we need corrections :(

↓

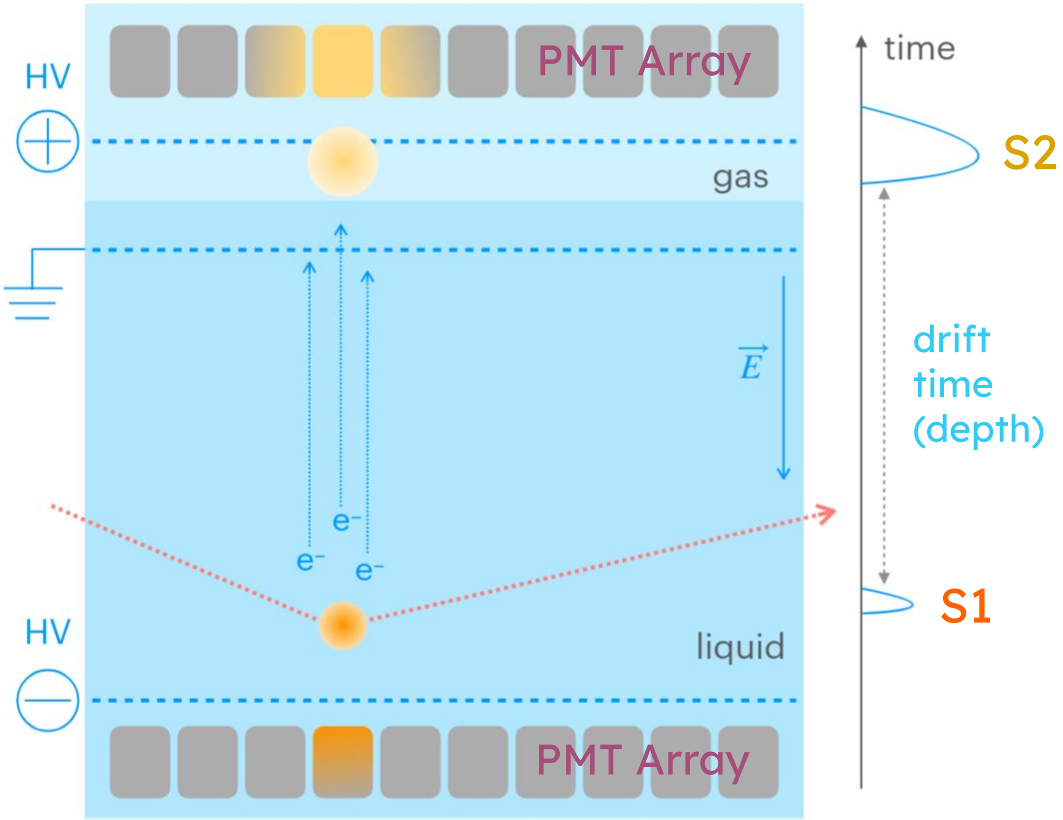
(x_{int}, y_{int}, z_{int})



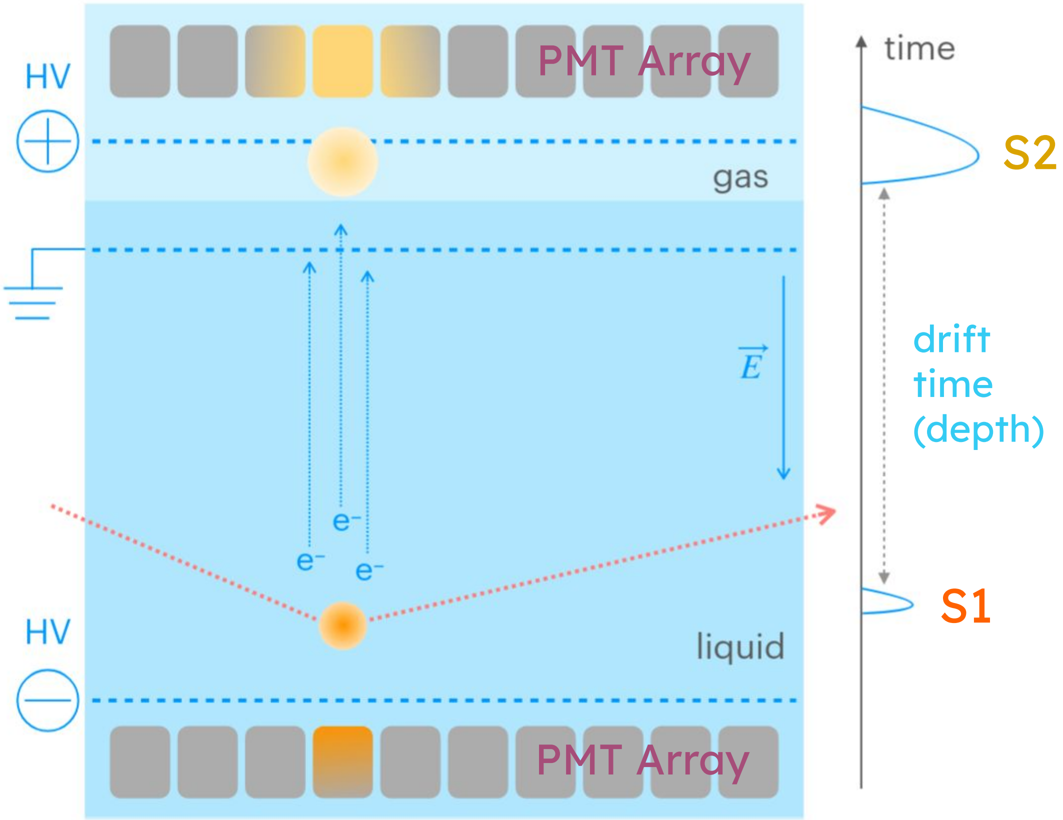
Simulation of drift field from XENONnT.
arXiv:2309.11996



The ionization electron starts from the first scintillation signal (S1) and ends at the second ionization (S2) signal.



The ionization electron starts from the first scintillation signal (S1) and ends at the second ionization (S2) signal.



The LXeTPC has a liquid layer and a gaseous layer of xenon, electric fields, and PMT arrays above and below the detector.

(Right) adapted from arXiv:2404.19524

We use a physics-informed continuous normalizing flow to model the electron movement due to the electric field.

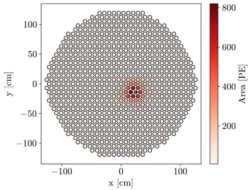
(hit_pattern, t_{S1}, t_{S2}, ...)

S2 position reconstruction

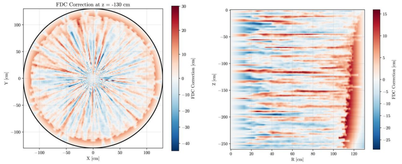
(x_{S2}, y_{S2}, drift_time)

corrections

(x_{int}, y_{int}, z_{int})

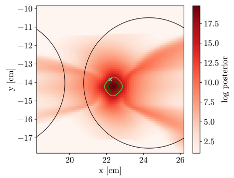


Maximum likelihood fitter (LZ) or train a dense NN (XENONnT)
Input: hit pattern
Output: (x_{S2}, y_{S2})

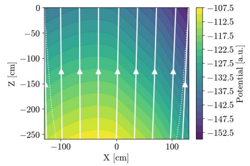


Traditional Data-Driven Field Distortion Correction Map (XENONnT and LUX)
Uniform monoenergetic calibration source
→ bin the data
→ find radial correction per bin
→ interpolate the map

Conditional normalizing flows for probabilistic position reconstruction
Based on work from Sebastian Vetter and Juehang Qin (paper and repo forthcoming)



Continuous normalizing flows for determining electric field lines for TPCs (this work!)
Paper and repo forthcoming, stay tuned!



A normalizing flow transforms a simple distribution into an observed distribution.

$$Z \sim \mathcal{N}(\mu, \Sigma)$$

(z_1, z_2) Bivariate normal

Normalizing Flow

$$X \sim p((x, y) \qquad)$$

(x, y)

A normalizing flow transforms a simple distribution into an observed distribution.

$$Z \sim \mathcal{N}(\mu, \Sigma)$$

(z_1, z_2)

Hit Pattern (c)

A conditional normalizing flow takes an additional condition.

(conditional) Normalizing Flow

$$X \sim p((x, y)|\text{hitpattern})$$

(x, y)

A normalizing flow transforms a simple distribution into an observed distribution.

$$Z \sim \mathcal{N}(\mu, \Sigma)$$

(z_1, z_2)

Hit Pattern (c)

A conditional normalizing flow takes an additional condition.

(conditional) Normalizing Flow

Find the most probable (x,y) given a hit pattern

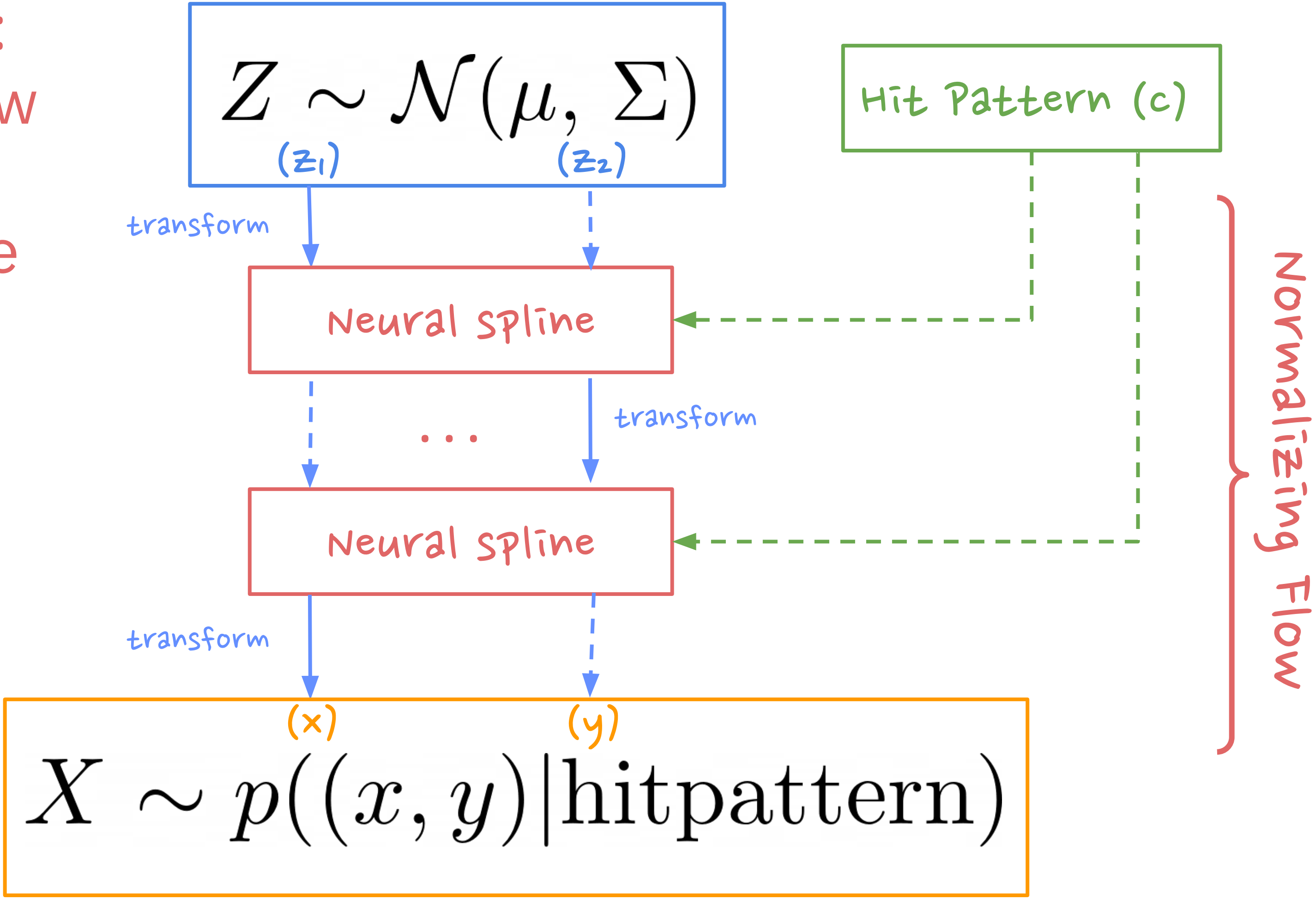
(inverse problem)

$$X \sim p((x, y)|\text{hitpattern})$$

(x,y)

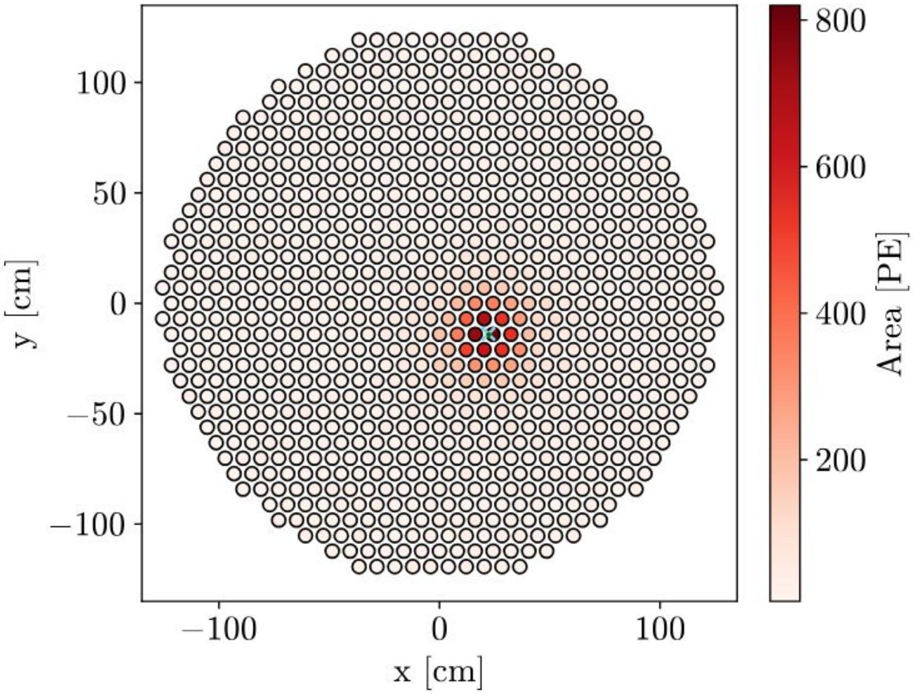
Architecture:
Coupling Flow
+
Neural Spline

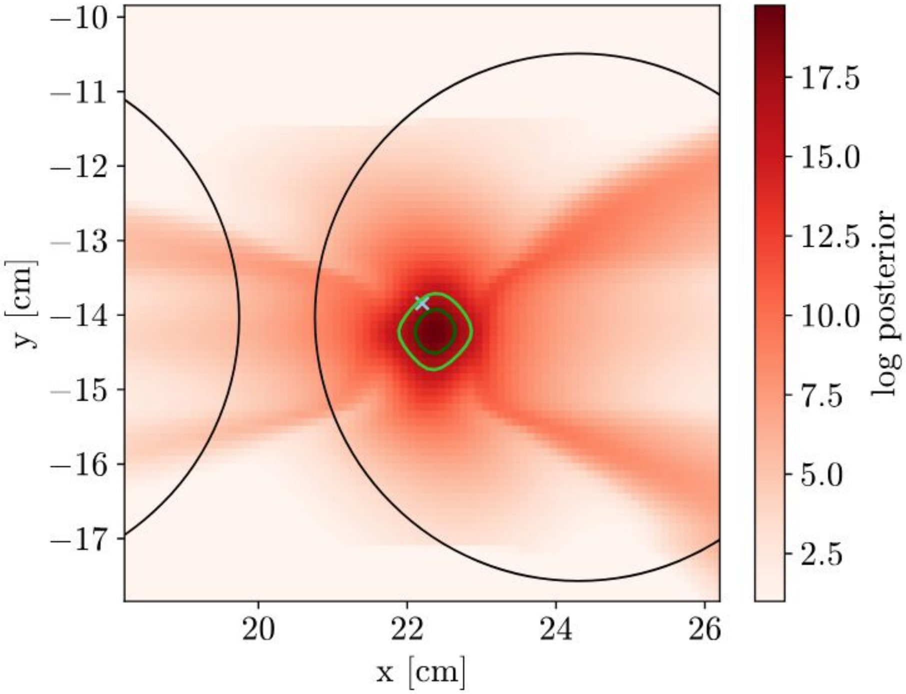
Series of
invertible,
flexible,
nonlinear
transformations



arXiv:1906.04032

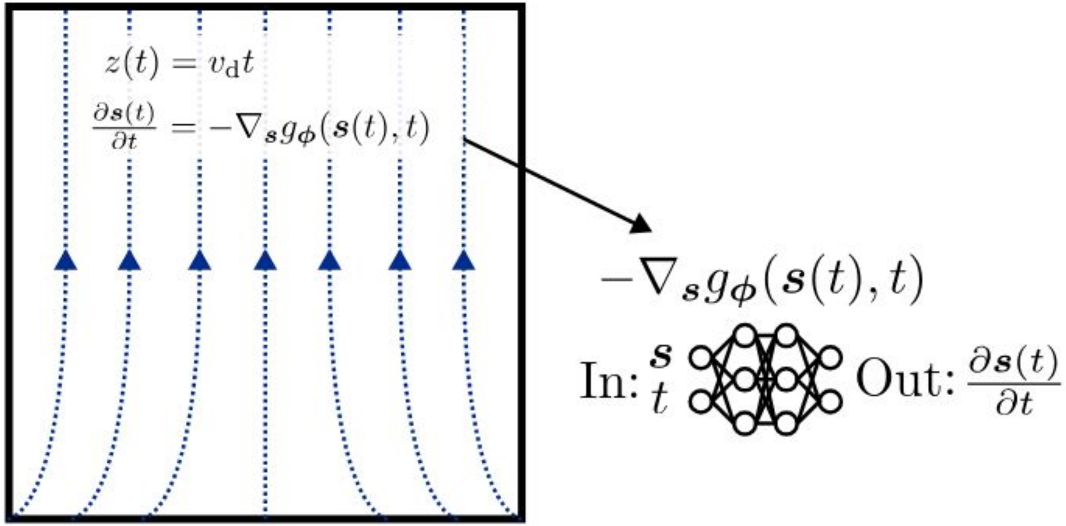
Density estimation to determine 68% and 95% exact contours on a given test dataset. An example of a reconstructed S2 contour looks like this.





Instead of modeling the electric field lines directly, the continuous normalizing flow learns a scalar potential map and then applies an ODE solver for each step.

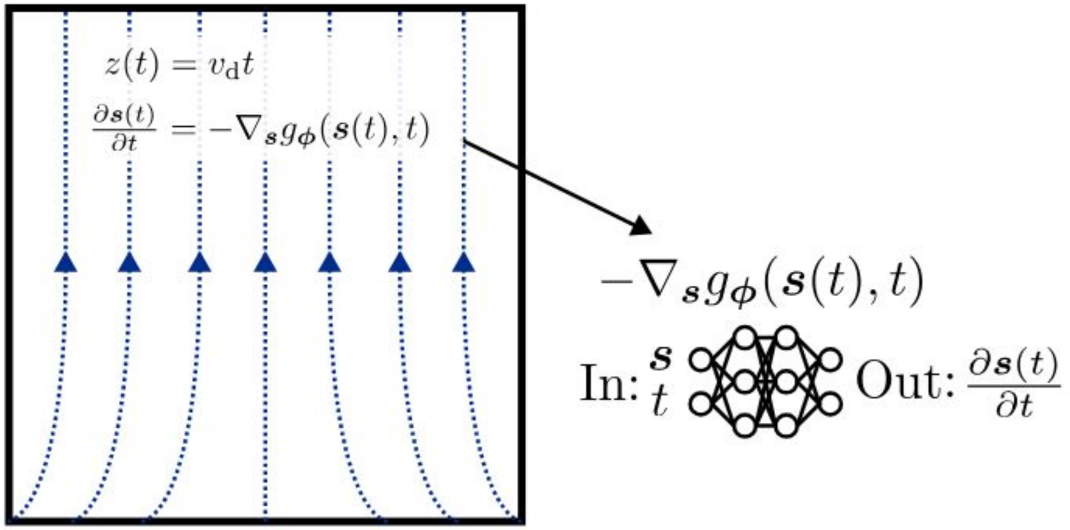
$$\begin{aligned}\frac{\partial \boldsymbol{s}(t)}{\partial t} &= -\mu_{\mathrm{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), v_{\mathrm{d}} t) \\ &\approx \boldsymbol{f'}_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t)\end{aligned}$$



Instead of modeling the electric field lines directly, the continuous normalizing flow learns a scalar potential map and then applies an ODE solver for each step.

$$\begin{aligned}\frac{\partial \boldsymbol{s}(t)}{\partial t} &= -\mu_{\mathrm{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), v_{\mathrm{d}} t) \\ &\approx \boldsymbol{f'}_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t)\end{aligned}$$

$$\boldsymbol{f'}_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t) = -\nabla_{\boldsymbol{s}} g_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t)$$

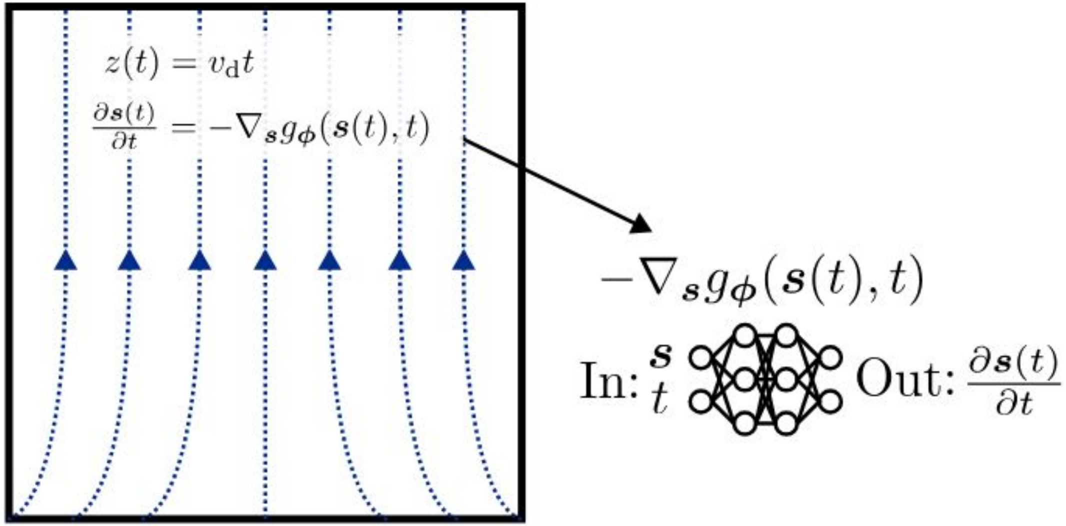


Instead of modeling the electric field lines directly, the continuous normalizing flow learns a scalar potential map and then applies an ODE solver for each step.

$$\frac{\partial \boldsymbol{s}(t)}{\partial t} = -\mu_{\text{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), v_{\text{d}} t)$$
$$\approx \boldsymbol{f'}_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t)$$

$$\boldsymbol{f'}_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t) = -\nabla_{\boldsymbol{s}} g_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t)$$

$$\frac{\partial \boldsymbol{s}(t)}{\partial t} = -\nabla_{\boldsymbol{s}} g_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t)$$

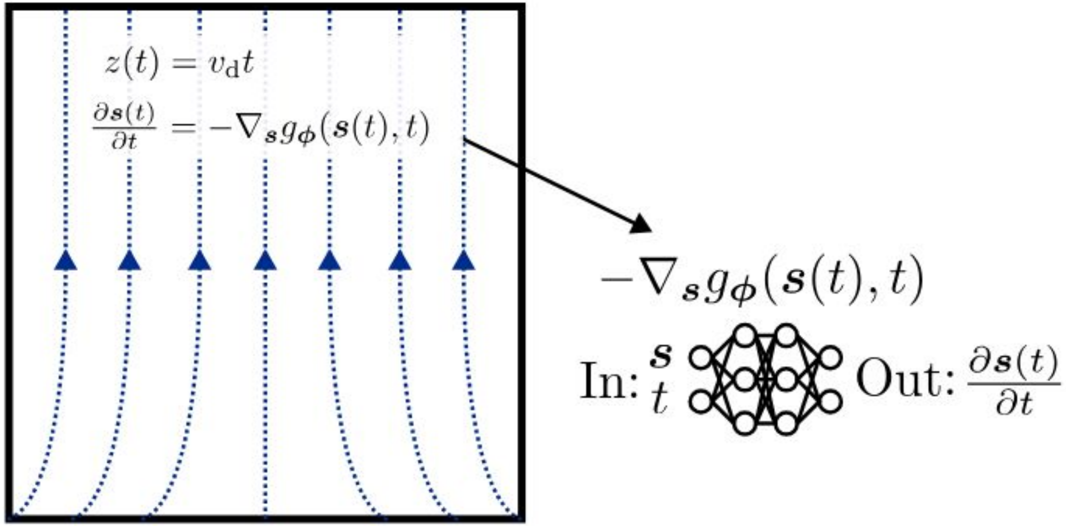


Instead of modeling the electric field lines directly, the continuous normalizing flow learns a scalar potential map and then applies an ODE solver for each step.

$$\frac{\partial \boldsymbol{s}(t)}{\partial t} = -\mu_{\mathrm{avg}} \boldsymbol{E}_{\boldsymbol{s}}(\boldsymbol{s}(t), v_{\mathrm{d}} t) \approx \boldsymbol{f'}_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t)$$

$$\boldsymbol{f'}_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t) = -\nabla_{\boldsymbol{s}} g_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t)$$

$$\frac{\partial \boldsymbol{s}(t)}{\partial t} = -\nabla_{\boldsymbol{s}} g_{\boldsymbol{\phi}}(\boldsymbol{s}(t), t)$$



We cannot directly measure the interaction vertex but we can measure a highly localized S2 hit pattern and the drift time.

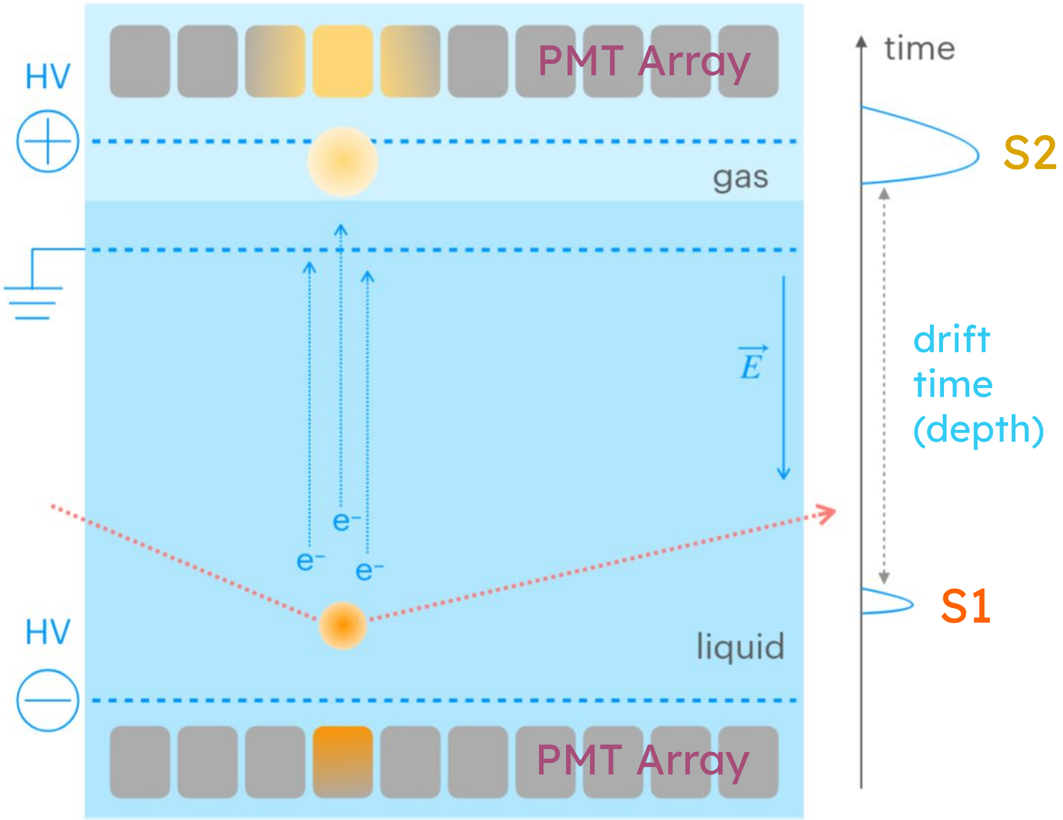
$(\mathtt{hit\_pattern}, t_{S1}, t_{S2}, \ldots)$

S2 position reconstruction

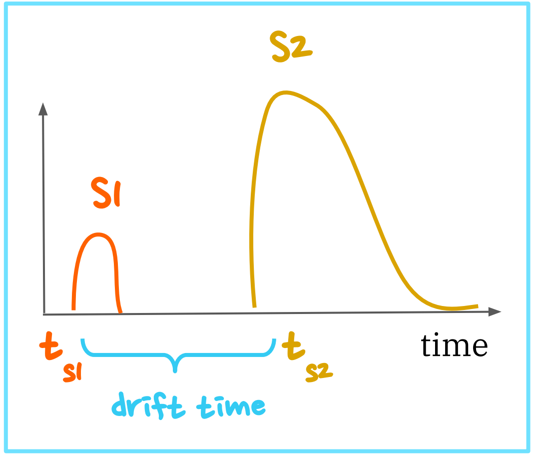
$(x_{S2}, y_{S2}, \mathtt{drift\_time})$

corrections

$(x_{int}, y_{int}, z_{int})$



We would like a machine learning way to build this electric field correction for position reconstruction.



An illustration of the sum waveforms of S1 and S2 signals.
Drawing by me

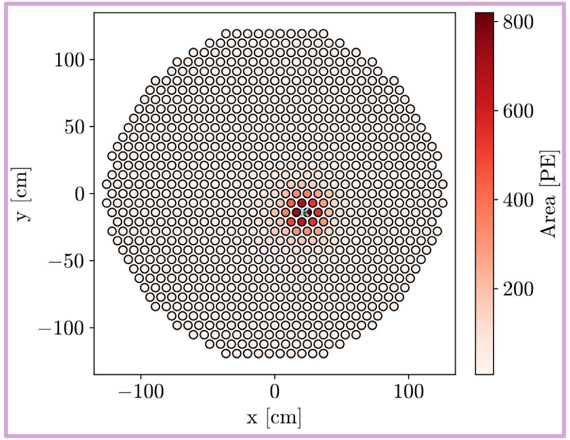
(hit_pattern, t_{S1}, t_{S2}, ...)

S2 position reconstruction

(x_{S2}, y_{S2}, drift_time)

E-field

(x_{int}, y_{int}, z_{int})



A simulated S2 hit pattern from the top array of PMTs.