Machine Learning in the MUSE Straw Tube Trackers

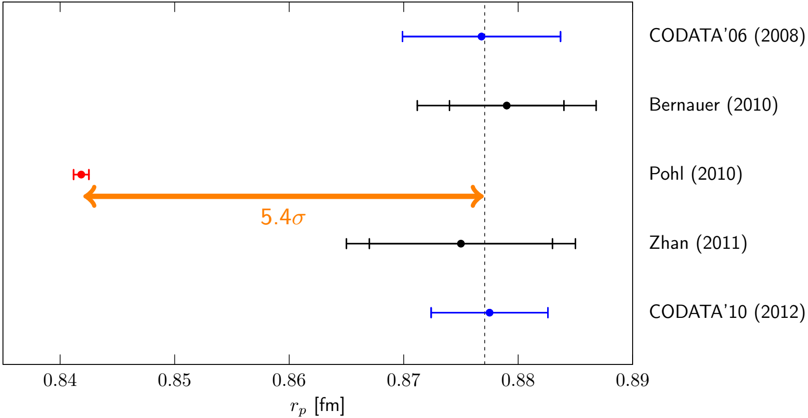
Kyle Salamone

Center for Frontiers in Nuclear Science, Stony Brook University

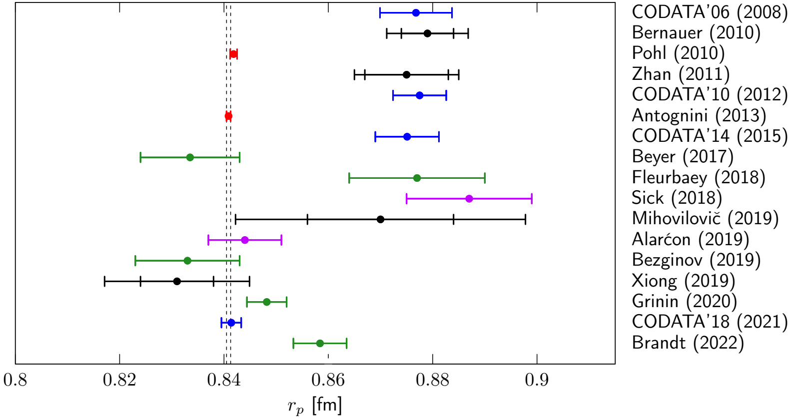
February 28, 2025

This material is based upon work supported by the National Science Foundation under NSF Grant PHY-2412703. The MUSE experiment is supported by the Department of Energy, NSF, PSI and the US-Israel Binational Science Foundation.

- 2010: CREMA collaboration measure Lamb Shift in muonic hydrogen
  - Results: $r_p = 0.84184 \pm 0.00067$ fm
- Average electron scattering measurement: ~ 0.877 fm



The Proton Radius Puzzle Today

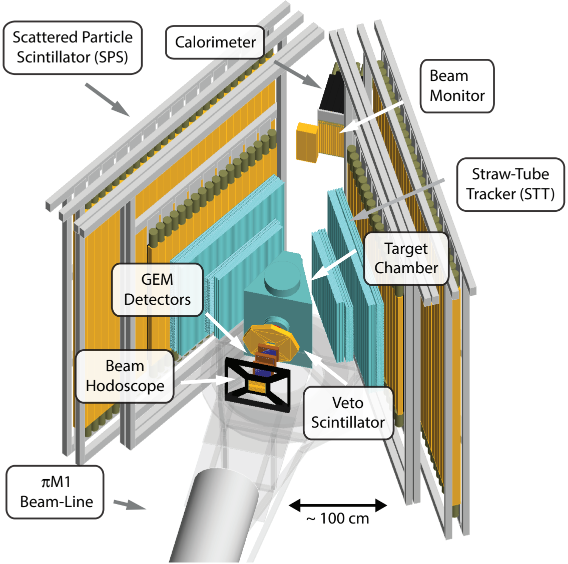


- Lepton universality?
- Radiative corrections (Two Photon Exchange (TPE))?
- Differences between spectroscopy and scattering?

The MUon Scattering Experiment (MUSE)

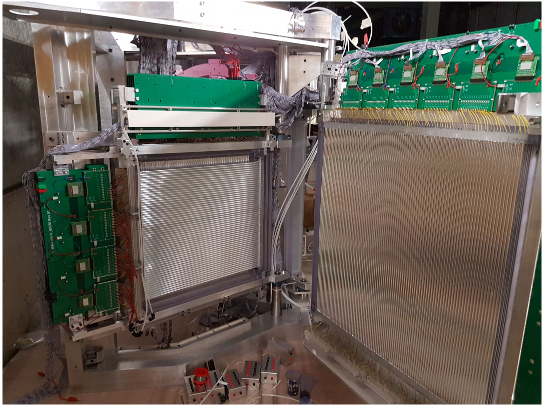
- The MUon Scattering Experiment (MUSE) was direclty inspired by the proton radius puzzle
- Goals:
  - Precision measurement of r_p via ep and μp scattering
  - Precision study of TPE in ep and μp scattering
  - Direct test of lepton universality
- Housed at the πM1 beamline at the Paul Scherrer Institute

- θ acceptance: $20 - 100^\circ$
- $\pi M1$ Beam Line:
  - p$\in 115, 160, 210$ MeV/c
  - Mixed beam of e, μ, π
  - Both polarities of particles!



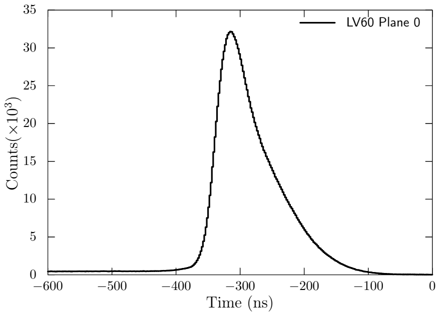
The Straw Tube Trackers (STT)

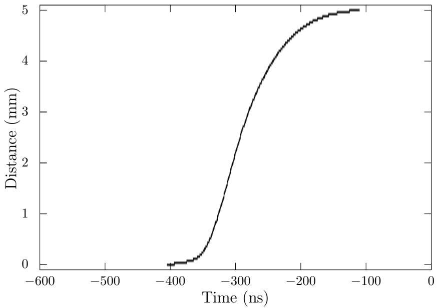
- Scattered particle tracking detector in MUSE
- Mirrored setup:
  - 20 planes of straws (10 horizontal, 10 vertical)
  - $\sim$ 3000 straws total!
  - Smaller front chamber, larger rear chamber
  - 5.1mm straw radius, 60 and 90 cm long
- Definition: set of 5 planes (so the 5 vertical horizontal, for example): "half chamber"



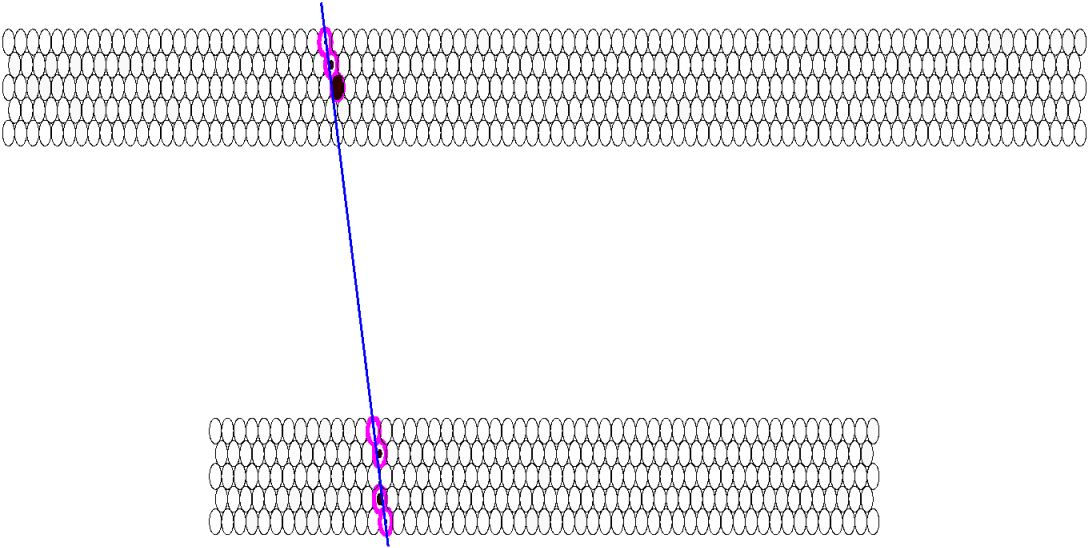
The Straw Tube Trackers (STT)

- "Standard" set of drift tubes
  - Measures time $\rightarrow$ distance, not (x, y, z)

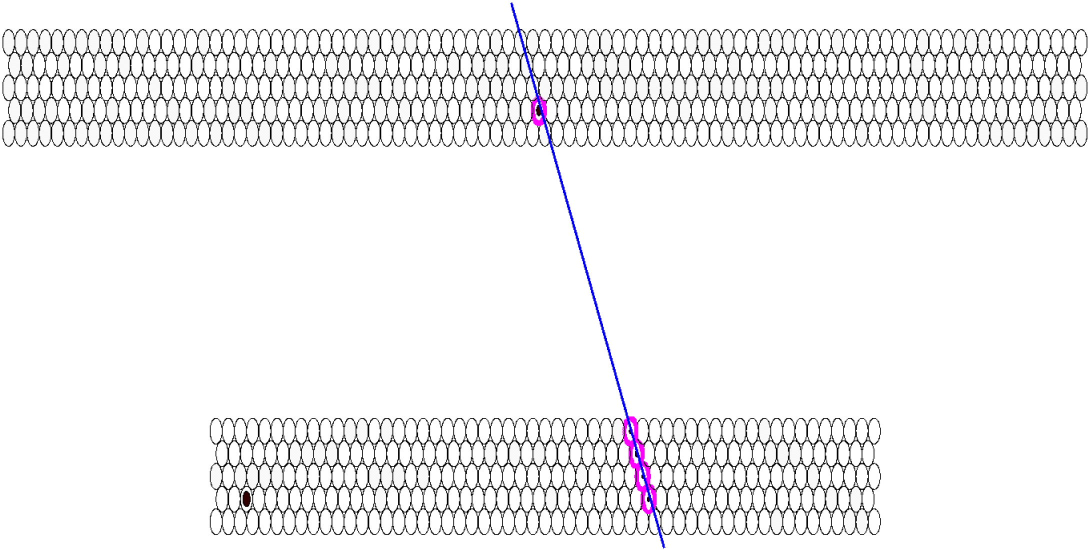




STT Tracking: Premise

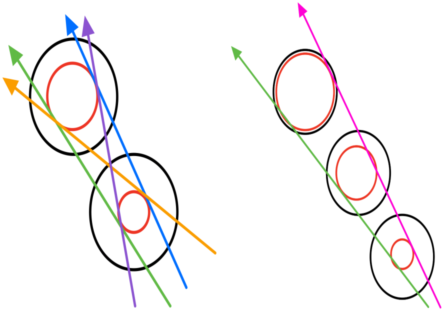


STT Tracking: Premise



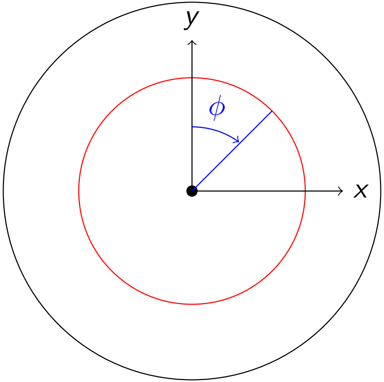
STT Tracking: Left Right Ambiguity

- Tracking is not straightforward: Left Right Ambiguity
- Makes χ^2 distribution in minimization complex
  - Many local minima for minimizer to get stuck in



Machine Learning Approach

- Idea: train a neural network to help resolve this ambiguity
- Few approaches to consider:
    - CNN to predict left/right alone
    - NN to predict ϕ in local straw frame
- Disclaimer: this is a work in progress!



Input Data

- For training: use Monte Carlo data
  - Generated 2 datasets: training and validation
  - Testing will be done later on run data
- Started with perfect radii, moved on to digitized/smeared
- Each "event" in the network:
  - At least 2 straws on a half chamber triggered by a primary particle
  - Direct input: 5 length 89 arrays; all 0 and radius of hit straws
  - Give the hit radius as well as the truth ϕ

```
plane,straw,real distance,phi

2,7,3.17937,1.34591

3,32,1.03686,1.19242
2,32,0.0207208,1.21275
1,31,1.61751,4.36256
0,31,3.17989,4.35121

4,21,1.10596,1.38557
1,21,2.0741,1.38562
3,21,2.11903,4.49986
0,22,2.25353,4.53246

0,54,1.94752,4.31264

1,0,3.01154,4.13951
0,1,2.2108,4.13119

3,12,0.93321,0.987719
4,12,0.252731,0.981482
1,13,1.97993,1.0019
```

NN Structure

- Structure of I/O:
  1. 5 arrays of length 89 (maximum straw in plane)
  2. All 0s (for loss masking) except for fired straws; these have their hit radii
  3. Concatenate all 5 arrays to form input
  4. Output: ϕ in radians, same concatenated form
- Training time for 100 epochs: ~ 12 minutes

```python
class LeftRightLearner(nn.Module):
    def __init__(self):
        super(LeftRightLearner, self).__init__()
        self.fc = nn.Sequential(
            nn.Linear(input_size, 2048),
            nn.BatchNorm1d(2048),
            nn.ReLU(),
            nn.Linear(2048, 1024),
            nn.BatchNorm1d(1024),
            nn.ReLU(),
            nn.Linear(1024, 256),
            nn.BatchNorm1d(256),
            nn.ReLU(),
            nn.Linear(256, input_size)
        )
    def forward(self, x):
        x = self.fc(x)
        return x % (2*torch.pi)
```

Custom Loss Function

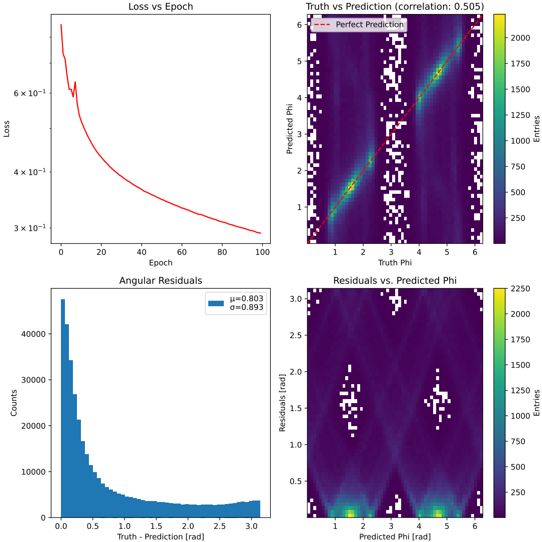
- Instead of ordinary MSE, defined my own
- Needed for angular differences wrapping around
  - For ordinary MSE: 1° and 359° have huge loss, but for this network it should be 2°

```
def MaskedCLLoss(truth, predicted):
    mask = (truth != 0)
    t, pred = truth[mask], predicted[mask]
    t, pred = t % (2*torch.pi), pred % (2*torch.pi)
    loss = torch.mean(1-torch.cos(t-pred))
    return loss
```

Training Tools

- Complex structure - use a learning rate scheduler
- Tested a few, chose "CosineAnnealingWarmRestarts"
  - Method of torch.optim.lr_scheduler
  - After T_0 epochs, resets the learning rate
  - This resetting happens every $T_0 + T_0 * T_{\text{mult}}$ epochs
  - Can set minimum learning rate
- Very large datasets - running using cuda

Results



Loss vs Epoch
Loss
Epoch
6×10^{-1}
4×10^{-1}
3×10^{-1}
Truth vs Prediction (correlation: 0.505)
Perfect Prediction
Predicted Phi
Truth Phi
Entries
Angular Residuals
μ=0.803
σ=0.893
Counts
Truth - Prediction [rad]
Residuals vs. Predicted Phi
Residuals [rad]
Predicted Phi [rad]
Entries

Thoughts

- NN seems to be progressing in right direction, but exists plenty of room for improvement
- Structure of NN is rather complex, potentially can be simplified
    - As well, may replace the hit radius on input with simply 1
- Bigger picture: implemtation in tracking code
    - Since we know r and $\phi \rightarrow (x, y)$ in local frame, can convert this to a hit in global frame
    - Write a simpler minimizer to take this information and fit a seed to these positions, give this seed to minimizers

Conclusions/Future

- Proof of concept for using NN to assist STT tracking showing proimise
- Will be tested on real data soon
- Future work:
  - Test on real data
  - Optimize network
  - Implement in tracking code
- Any comments/suggestions are welcome!

Form Factor Discrepancies