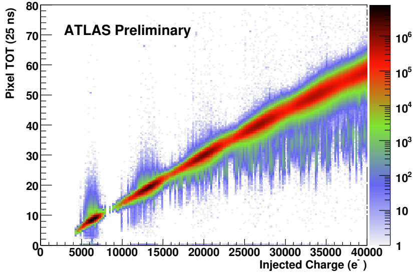
Neural-network measurement calibration

Louis-Guillaume Gagnon (UC Berkeley)

ACTS developers workshop 2023
2023/10/11

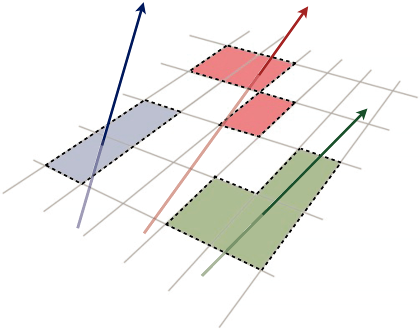
Introduction: Pixels

- Running example today: Silicon Pixel Detector
- Ionisation due to incident charged particle
  $\rightarrow$ measured voltage in individual pixels
- Pixel chip records the time-over-threshold
- With calibration, convert to deposited charge
- Typically measured directly with known charge injection
- Outside of scope of ACTS – handled by experiments
  - not the calibration that's covered today!



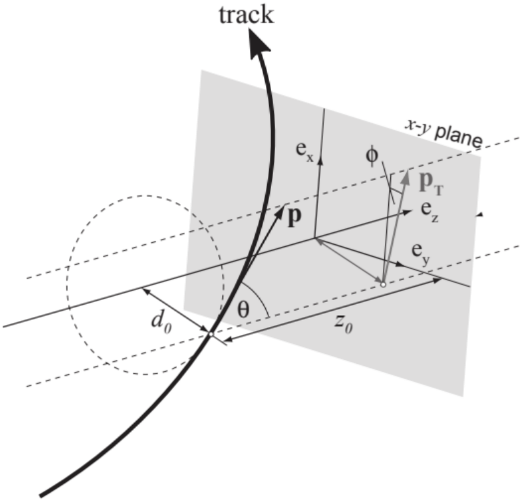
Introduction: Measurements

- From Si pixel detector: get location and charge of above-threshold pixels
- Beginning of ACTS Core scope: Clusterization
- Connected component analysis $\rightarrow$ charge clusters
- In ACTS: Hoschen-Kopelman algorithm
  - "Implicit" raster scan over pixels
  - Use disjoint set forest to keep track of cluster assignment
- Initial position estimation from measurement: Currently outside of scope of ACTS Core!
  - In General, need knowledge of readout geometry
- In ACTS examples, simple strategy:
  - Position estimate: Charge-weighted average of pixel center positions
  - Uncertainty estimate: Pixel width $/\sqrt{12}$



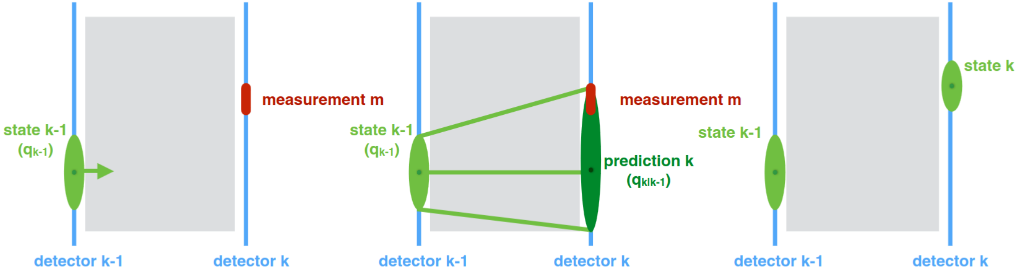
Introduction: Kalman Filter

- ACTS Track state model: $(d_0, z_0, \theta, \phi, q/p, t)$
  - with associated covariance
- Estimated with measurements from detector
- E.g. for pixel detector: $m = (x, y)$
  - with associated covariance, usually diagonal
- Track state incorporates measurements via Kalman Filter formalism
  - Start from track seed parameters
  - Predict parameters at next surface
  - Search for matching measurements
  - Kalman update stage: Update track state using matching measurement
  - Repeat until no more surfaces
- Nucl.Instrum.Meth.A 262 (1987) 444-450



Introduction: Kalman Filter

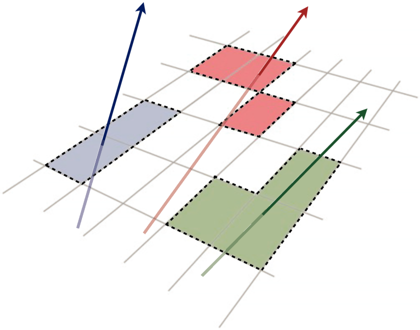
- At a glance:



[2105.01796]

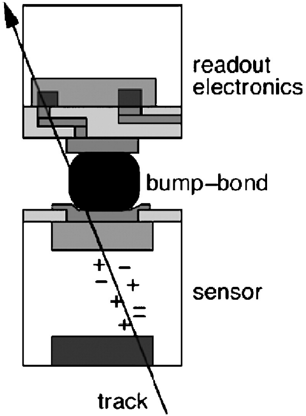
- Left: Have a track state at layer $k-1$ and a measurement on layer k
- Center: Using known track state and its covariance, Predict track state at layer k
- Right: Obtain track state at layer k by updating the prediction with the measurement

- From pixel detector, obtained measurements $m = (x, y)$
  - $(x, y) =$ charge-weighted cluster center
  - $(\sigma_x, \sigma_y) =$ pix. width / $\sqrt{12}$
- Possible to improve:
  - Take direction into account
  - Do fancier shape analysis
  - ...
- Measurement calibration paradigm: Apply corrections to estimated measurements during Kalman update stage
  - Simple scale-and-offset schemes
  - ATLAS: "Analogue clustering", NN-based clustering
  - ...many other possibilities



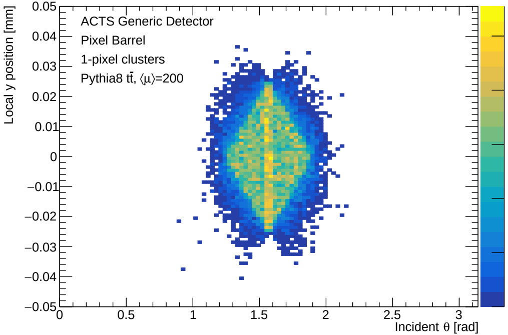
Simplest Possible Example: Single-pixel measurement

- Primarily rely on shape analysis to constrain position
- Edge case: 1-pixel clusters, "no" shape information
- However: Angles of incidence give some constraint!
  - $\approx 90^\circ$ crossing: Anywhere on surface
  - $\rightarrow 0^\circ$ crossing: Near center (else, ≥ 2 pixel)
- N.B. position defined at middle of Si bulk, by convention



Simplest Possible Example: Single-pixel measurement, longitudinal position

- Single-pixel clusters offer simple example of interplay between cluster shape, crossing angles, and position
- Clear relationship between $\sigma(\mathrm{pos})$ and angle
- $\theta \approx \pi/2$ (head-on): σ largest
- $\theta \ll \pi/2$: σ smallest
- Intuition: If $\theta \ll \pi/2$ AND position not near center: high likelihood of having ≥ 1 pixel cluster!
- $\implies$ If can estimate crossing angle, can assign "correct" irreducible uncertainty to measurement



Measurement Calibration with Neural Networks

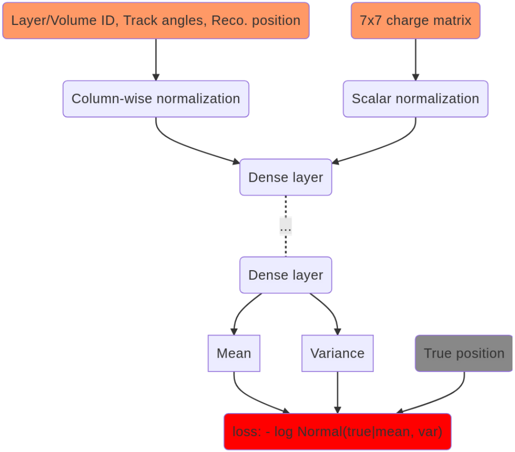
- MDN $\equiv$ Mixture Density Network
- i.e. any neural network trained to output parameters of a gaussian mixture
- Model output: parameters π_i, μ_i, σ_i such that:

$$P(Y|X) \sim \sum_i \pi_i(X)\mathcal{N}(Y|\mu_i(X), \sigma_i(X))$$

  - X is set of variables describing a measurement (e.g. charge, volume/layer, angles of incidence)
  - Y is true crossing position in Si bulk (ground truth)
  - $\pi_i(X)$: Prior probability for i-th component (if using ≥ 2 components)
  - $\mu_i(X)$: Calibrated position estimate (Supervised learning)
  - $\sigma_i(X)$: Uncertainty estimate (Unsupervised learning)
- If using single component, model is a simple normal distribution
- Trained using probabilistic programming paradigm: loss is directly $-\log P(Y|X)$
- At runtime, use $\mu_i \pm \sigma_i$ corresponding to highest π_i as position estimate
- This method naturally generalizes to clusters with ≥ 2 particles
- Method used by ATLAS collaboration for pixel measurement calibration
  - See e.g. ATL-PHYS-PROC-2019-082

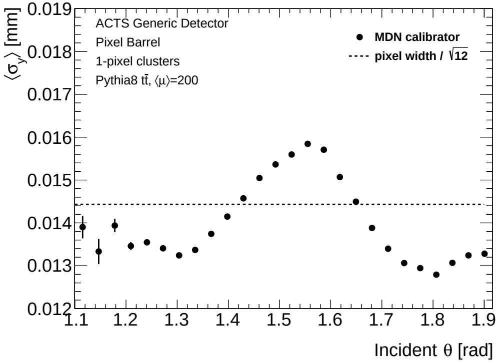
Possible neural network architecture

- Example Architecture to work with NeuralCalibrator in ACTS Examples
- Input $\rightarrow$ NN $\rightarrow$ Mean, Variance: Can be any neural network
- For proof-of-concept: simple `tensorflow.keras` dense network
- Loss: IndependantNormal layer from `tensorflow_probability`
- "public" example coming soon™
  - Not one-size-fits-all detector-agnostic network
  - Rather, reference implementation + "how-to-train" documentation



Measurement Calibration with Neural Networks: in action

- Clear relationship between $\sigma(\mathrm{pos})$ and angle
  - Stronger constraint at large angles
  - Weaker constraint for head-on particles
- "Head-on" variance $>$ pixel width / $\sqrt(12)$: Charge drift from neighboring pixels (?)
- $\sigma_{x/y}$ are model-estimated uncertainties, not residuals



Calibration interface in ACTS

- The ACTS tracking toolkit contains Kalman Filter-based track finding & fitting algorithms
- Calibrations can be applied on-the-fly during track finding / track fitting
- Interface implemented using template-based delegation:

```
class KalmanFitterExtensions {
  using Calibrator = Delegate<void(const GeometryContext&, const CalibrationContext&,
                                    const SourceLink&, TrackStateProxy)>;

  /// The Calibrator is a dedicated calibration algorithm that allows
  /// to calibrate measurements using track information, this could be
  /// e.g. sagging for wires, module deformations, etc.
  Calibrator calibrator;

  ...
};
```

- Calibrator class acts directly on track state proxy, which holds the current measurement
- Dynamic geometry effects and intra-run calibration changes encapsulated via contextualization

Calibration interface in ACTS Examples framework

- See `ActsExamples/EventData/MeasurementCalibration.hpp`
- The ACTS Core calibration interface does not directly take measurements
- Previously: Calibrators would be instantiated for each event with vector of measurement in constructor
- Fine for calibrators with "trivial" initialization, not fine for more complex cases
- Now have access to a different approach:
  - `MeasurementCalibrator` base class that accepts vector of measurements in its `calibrate` method
  - `MeasurementCalibratorAdapter` wrapper that binds a vector of measurements to a calibrator
- Calibrator can be instantiated once, outside of event loop
- Adapter instantiated for each event, with trivial initialization
- Adapter has a `calibrate` method that conforms to the ACTS Core interface
- Uses of this interface in the Examples:
  - ScalingCalibrator
  - NeuralCalibrator (Uses ONNX plugin)

Conclusion

- Measurement Calibration: Correcting the measurement positions & errors on-the-fly during track finding & track fitting
- The ACTS Kalman Filter includes efficient template-based interface to measurement calibration
- Different examples are provided: Simple ScalingCalibrator, Fancy MDN-based NeuralCalibrator
- Future plans:
  - Provide documentation and tutorials for the interface and the examples
  - Explore more calibration methods (e.g. ATLAS "Analogue Clustering")
  - Implement ATLAS-inspired dense environment calibration (Cluster splitting, positions for ≥ 2 particles, ...)