4D-STEM Requirements Specification

Ciaran Welsh / SEM Team

August 14, 2025

Contents

1 Introduction

This document defines the implementation-independent software requirements for 4D-STEM users. It is a reference for software and application engineers as well as the business side of ASI. The document should be considered 'living' in that the information should be updated with pertinent information as the project evolves.

4D-STEM (Four Dimensional Scanning Transmission Electron Microscopy) is a microscopy technique where a focused electron beam is scanned across a sample, producing a 2D diffraction pattern at each position. The resulting four-dimensional dataset, with two spatial and two reciprocal-space dimensions, enables detailed analysis of structure, strain, and other material properties. The advantage that Timepix3 (and Timepix4) offer such an application are derived from the event-driven nature of the detector; timepix detectors are event driven and therefore record each hit independently from other pixels, and therefore avoiding some of the frame-based limitations and offering greater performance. With tpx3 in 4D-STEM we can get small dwell times, improved dose efficiency for beam-sensitive samples, reduced data volume and supports real-time analysis with lower memory and bandwidth requirements.

From a business perspective, the primary project goal is to sell more ASI detectors by embedding them into the 4D-stem solution. The 4D-STEM component adds considerable value to the detector and therefore makes this a worthy enterprise for ASI. Our goal is to build a full 4D-STEM solution, including both hardware and software components. Importantly we design for and include the next generation of timepix (Timepix4) into the design from day 1. We might not be able to support the full rate timepix 4 from the begining but with appropriate modularity in the design we should be able to drop in higher performance replacements for performance critical components (such as versions that run on the GPU).

To date, ASI have focused their efforts on building this feature into Accos and Serval, but with difficulties. This document is not intended to be a review of that work, but suffice to say that the existing solution is not an ergonimic experience for neither the user nor the developer. The proposal outlined in this document is intended as a replacement to the 4D-STEM corners of Accos and Serval.

In the following sections we describe the system we intend to build from a high level perspective. We describe the envionment the system needs to run, including software and hardware dependencies and we outline the needs of the end user that this software aims to solve. We outline functional requirements and the architectural characteristics that developers need to design for, based on these needs.

2 Overall Description

The primary function of this software is to enable users one integrated workstation for the control of their detector and scan generators. The software needs to be one cohesive unit that enables scientists to conduct their experiments and collect data for further analysis. It is the first tool in their chain of tools that delivers them results for publications. As such this software needs to deliver ergonimic workflows for the evaluation and acquisition of their sample. It needs a performant live image stream to provide microcopists with enough information to evaluate a section of sample and interoperability with other tools that enable further study of their data, including LiberTEM, HyperSpy and py4dStem. We do not focus on specialised downstream analysis techniques such as ptychography.

The software must be suitable for execution on a single workstation. Serval is the primary mechanism for controlling the detector and we need multiple mechanisms of control for a choice of scan generators to reach a larger audience. From the user perspective however, the new software is the one over-riding central point of control for all the tools necessary to conduct their experiment. A general overview of the application's position in the broader ecosystem is shown in fig. 1.

External connections are:

- **Serval**: REST API for detector configuration and TCP stream for event data.
- **Scan generator**: Controlled via vendor-supplied shared library (FFI), gRPC or another system of communication.
- **Local storage**: High-throughput NVMe drives for data capture.

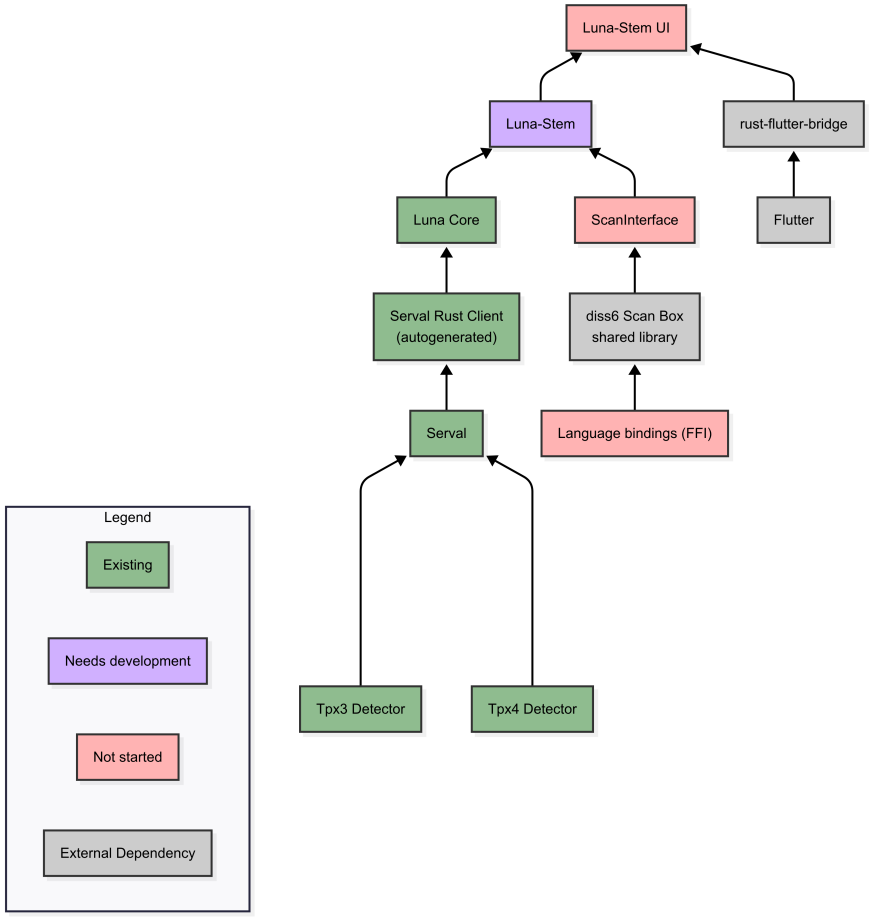

Figure 1: Overview of the application in relation to other tools

2.1 User Classes and Characteristics

The primary users are 4D-STEM scientists, who also operate the microscope. Users range from computationally literate power users to those without much experience. Both users are mainly focused on scientific outcome rather than on the processing required to get them there. The software abstracts data acquisition and visualisation details so that no programming skills are required for them to do just that.

2.2 Definitions and Acronyms

1. ¡expand and define¿
2. CBED:
3. vADF:
4. iDPC:
5. iCOM:
6. Diffraction bands: the amount of diffraction an electron experiences is related to the charge of the atoms inherent in the sample (please validate this statement). A typical sample therefore has characteristic diffraction patterns which can be selected using an annulus shape.

2.3 Operating Environment

The application supports Windows, Ubuntu, and macOS at launch; other Linux distributions can be built on request. Our primary system is ubuntu and several aspects of our tooling are already optimized for ubuntu. We therefore aim for performance targets primarily on ubuntu, and only secondarily on the other operating systems. The software shall run on a high-performance lab workstations with specifications sufficient for sustained high-rate data capture and processing. GPU acceleration is planned for future versions but is not required initially. The system operates on localhost only, with all components running on the same machine. Storage must sustain the highest practical write speeds for the selected output format.

2.4 User Needs

todo: to better pin down what the system need to do we could turn this into a list of user stories.

1. **Seamless integration with Serval for detector control whilst being exposed to only what they need**
   Rationale: Serval must be used for detector control but minimizing the user interaction with Serval reduces the learning curve and enhances usability.
2. **Real-time visual feedback during scans**
   Rationale: Enables rapid navigation to regions of interest and immediate quality assessment. Also prevents wasted beam time.
3. **Ability to run essential offline analyses inside the application, such as offline image reconstruction methods**
   Rationale: Enables quality assessment decisions without exporting large datasets to external tools.
4. **Acquire continuous or multi-frame ('Z-stack') scans**
   Rationale: Lets users monitor radiation damage frame-by-frame and perform statistics based on multiple measurements in a region.
5. **Flexible detector masking (hardware or software) to select pixels for subsequent analysis.**
   Rationale: Enables selection of diffraction bands for online and offline image reconstruction algorithms.
6. **Switchable scan modes to balance performance and user feedback trade-offs**
   Rationale: Allocation of computational resources should be allocated differently depending on whether the user is searching for interesting features or whether they have already found them and want to acquire data for further analysis.
7. **Export raw *or* processed data in open formats (e.g. SparseArray, HDF5, Zarr) and where appropriate, image data**
   Rationale: Guarantees images can be saved and interoperability with downstream analysis suites such as LiberTEM, HyperSpy, and py4DSTEM.
8. **Real-time observability of acquisition-pipeline health (hit-rate, processing load, synchronization)**
   Rationale: Gives user vital feedback that can be used to understand what is happening in the pipeline. This includes amounts of dropped data, per chip level hit rates and data throughput/latency rates.
9. **Dedicated 4D-STEM interface separate from the generic detector UI**
   Rationale: Build with a 4D-stem first mentality. Makes software 4d-stem centric, minimises cognitive load on our users needing to compete with more general use cases of the timepix3 detector and prevents accidental changes to routine detector operations. Separation of concerns.
10. **Immediate and clearly organized access to scan-generator controls**
    Rationale: Scan controls are some of the most important in 4D-STEM, they should be prevalent and obvious in the user workflow for better user experience, operating on the principle of least clicks to control your scan.

3 System Features and Requirements

3.1 Functional Requirements

3.1.1 Serval Integration

Serval is the control interface between the application and the detector.

1. Must **REQ-SRV-01 – Connect or spawn Serval.** On startup, the system shall either (a) connect to a reachable Serval instance at the configured host:port, or (b) spawn Serval as a child process whose lifetime is tied to the 4D-STEM UI.
2. Must **REQ-SRV-02 – Version negotiation.** On initial connect, the system shall retrieve Serval's API version and compare it to the supported range. If out of range, the UI shall warn the user.
3. Must **REQ-SRV-03 – State subscription (push).** The client shall subscribe to Serval state changes (e.g., detector {Disconnected, Connected, Idle, Running}). Won't The system shall not use periodic polling to keep up to date with serval connectivity status.
4. Must **REQ-SRV-04 – Detector connection sequence.** Upon receiving *detector_connected* signal, the system shall: (1) upload DACs, (2) upload pixel configuration (BPC), (3) set data destination, (4) load saved detector settings (e.g. bias) for this detector chipboard Id. Failures stop the sequence and surface a recoverable error with instructions on how to resolve the issue.
5. Must **REQ-SRV-05 – Serval Command error model.** All Serval command failures shall be presented to the user as human readable message and remediation hint.
6. Must **REQ-SRV-06 – Connectivity loss during acquisition.** If the control connection to Serval drops during an active acquisition the acqiusition is considered invalid. The system shall detect connection loss and abort the acquisition gracefully.
7. Must **REQ-SRV-07 – Logging and correlation.** Every Serval command, response, state change, and error shall be logged in the serval http log stored per workspace
8. Must **REQ-SRV-08 – Transport keepalives.** The client shall use protocol keepalives to detect dead connections without polling for state.
9. Must **REQ-SRV-09 – Endpoint configuration.** Serval host, port, and transport addresses shall be configurable in the UI; changes take effect on "accept" button press.
10. Won't **REQ-SRV-10 – Equalization in client.** The client shall not perform detector equalization; that function remains in Accos.
11. Won't **REQ-SRV-11 – Detector orientation in client.** The client shall not manage detector orientation.

3.1.2 Online Image Reconstruction

1. Must **REQ-OIR-01 – Real-time vADF reconstruction.** Raw event data received from Serval via TCP shall be processed in real time into images using the vADF algorithm. Processing is performed by *luna-stem*/*luna-core* and displayed in the 4D-STEM UI. For selection of pixels that contribute to final vADF image, an annular ROI shall be used (item 16)

   .
2. Must **REQ-OIR-02 – Preview non-interference.** Preview must not reduce performance to such a point that data integrity is compromised.
3. Should **REQ-OIR-03 – GPU-accelerated vADF.** Provide a GPU-accelerated implementation of vADF on supported hardware. Especially when targeting tpx4 (in future) which has x5 the performance requirements of a tpx3 turbo, a GPU backend should be built

3.1.3 Offline Image Reconstruction

1. Must **REQ-OFR-01 – Single CBED display.** When a user selects a probe position, the system shall display the corresponding CBED diffraction pattern from disk.
2. Must **REQ-OFR-02 – Average CBED (all).** The system shall compute and display the Average CBED over all probe positions in the scan.
3. Must **REQ-OFR-03 – Offline vADF.** The system shall recompute vADF from stored data by summing intensities within the selected detector region for each probe position. The selection ROI shall be annular.
4. Must **REQ-OFR-04 – iCOM.** Provide offline iCOM reconstruction on supported datasets. Performance must be "reasonable" but not heavy duty, like the online algorithms
5. Should **REQ-OFR-05 – iDPC.** Provide offline iDPC reconstruction on supported datasets. Performance must be "reasonable" but not heavy duty, like the online algorithms

6. Must **REQ-OFR-06 – Asynchronous execution + progress.** Offline operations shall run asynchronously with progress reporting and possibility of user cancellation. Partial outputs are marked incomplete with a warning and message.
7. Could **REQ-OFR-07 – Random-access readiness.** If needed for performance, maintain an index that maps probe position to data blocks/offsets next to processed data output to facilitate fast lookup and retrieval of data for building images (such as CBED).
8. Won't **REQ-OFR-08 – Ptychography (offline).** Offline ptychography is out of scope.
9. Won't **REQ-OFR-09 – 4D-STEM EELS (offline).** Offline 4D-STEM EELS is out of scope.

3.1.4 Z-stacking Frames

1. Should **REQ-ZST-01 – Fixed-length Z stack.** When the user sets $Z > 1$ and presses *Start (Acquire)*, the system shall execute exactly Z full probe-grid scans. Every detector event shall be assigned to frame and probe position.
2. Should **REQ-ZST-02 – Continuous Z mode.** When Continuous Z mode is selected, the system shall repeat full probe-grid scans until the user issues *Stop* or an automatic stop condition occurs (e.g., disk guard). Frames in continuous mode shall use a monotonically increasing *frame_id* $\in \{0, 1, 2, \dots\}$.
3. Should **REQ-ZST-03 – Frame output writing.** Frame data shall be written to disk in the user-selected format(s).

3.1.5 Output and Raw Data Collection

1. Could **REQ-OUT-01 – Acquire preview policy.** If necessary, and depending on the performance characteristics of the system (yet to be determined), the final acquisitions shall run with preview disabled by default. If explicitly enabled, the user shall be warned before they acquire that preview + acquisition on the same computer can lead to data loss.
2. Must **REQ-OUT-02 – Raw data immutability.** Raw event data shall be written to disk exactly as received from Serval (ordering and content preserved), with no corrections applied in-line by the client or Serval.
3. Must **REQ-OUT-03 – Corrections policy.** Corrections required for scientific correctness (e.g., tram-lane and PLL corrections) shall be applied only in reconstructions (online/offline), never to raw. Correction parameters are retrieved from Serval (or calibration files) and recorded in provenance.
4. Won't **REQ-OUT-04 – Timewalk correction in raw.** Timewalk correction shall not be applied to raw event data. The time resolution is usually not small enough to have a huge impact.
5. Must **REQ-OUT-05 – Sparse event export (LiberTEM).** Provide a sparse event export compatible with LiberTEM (indices/values plus shape/coords).
6. Should **REQ-OUT-06 – Zarr export.** Provide a Zarr export for 4D data (e.g., $[H, W, Q_x, Q_y]$) with chunking and compression per DATA-FMT; write online if feasible, else offline converter.
7. Should **REQ-OUT-07 – HyperSpy HDF5 export.** Provide HyperSpy-compatible HDF5 export for reconstructed data; write online if feasible, else via offline converter.
8. Must **REQ-OUT-08 – Image export (2D).** Any 2D image produced (e.g., vADF, CBED, iCOM maps) shall be exportable to TIFF and PNG; 16-bit TIFF supported; metadata embedded where format allows.
9. Could **REQ-OUT-09 – ROI-based export.** For data reduction, users may export subsets (ROIs in real space or diffraction space) to supported formats without duplicating full datasets.
10. Could **REQ-OUT-10 – Offline format converters.** Provide CLI/GUI converters among supported formats
11. Must **REQ-OUT-11 – Directory layout and naming.** Acquisitions shall use a standardized directory layout and naming convention. Users can modify their pattern with a selection of template strings. Datetime, run ID, sample ID are examples.
12. Could **REQ-OUT-12 – Compression and chunking.** For formats supporting it (Zarr/HDF5), chunking and compression shall be implemented for offline use.

3.1.6 Pixel Masking

1. Must **REQ-PMK-01 – Software masking.** The system shall support software inclusion zones or 'mask' defined on the detector pixel grid (rectangular, circular, and annular at minimum). Software masks are applied in image reconstruction; raw data remain unchanged.
2. Could **REQ-PMK-02 – Hardware masking.** The system shall support hardware masking by uploading a pixel configuration via Serval.
3. Must **REQ-PMK-03 – Mask mode selection.** If both hardware and software masking are available, the UI shall allow selecting the active mode; switching is permitted only when idle.
4. Must **REQ-PMK-04 – Mask visualization.** Masks shall be visibly overlaid on the viewport; hardware and software masks use distinct visual styles.
5. Must **REQ-PMK-05 – Autoload masks.** When loading an existing dataset/measurement, previously saved masks shall autoload and appear in the UI.

3.1.7 Acquisition Modes

1. Must **REQ-ACQ-01 – Search mode.** Search mode shall render real-time preview (vADF) and, by default, shall not persist raw events or reconstructions.
2. Must **REQ-ACQ-02 – Acquire mode.** Acquire mode shall persist raw data data to disk in Sparse matrix format. Preview is disabled by default; if enabled, it must not interfere with data acquisition.
3. Must **REQ-ACQ-03 – Mode transitions.** Mode selection shall be explicit and visible; mode changes are blocked while an acquisition is active.
4. Should **REQ-ACQ-04 – Parameter locking in Acquire.** On *Start (Acquire)*, scan parameters and masks are locked for the duration of the run.

3.1.8 Scan Generator

1. Must **REQ-SCN-01 – Backend selection.** The system shall support selecting a scan-generator backend from a configured list; unavailable backends appear disabled.
2. Must **REQ-SCN-02 – Start-time synchronization.** Detector acquisition start and scan start shall be synchronized at the start-of-scan boundary. How do we measure this?
3. **REQ-SCN-03 – ROI/geometry translation.** Scan generator shall accept ROI-derived geometries and produce matching probe grids for raster scan.
4. Must **REQ-SCN-04 – Graceful stop.** On *Stop*, the scan generator shall honor the 'stop now' and 'stop at frame boundary' semantics from Z-stacking.

3.1.9 Observability and Pipeline Health

1. Must **REQ-OBS-01 – Metric definitions.** Performance metrics (hit-rate, throughput in bits/s, ingest→present latency) shall be computed per Appendix
2. Must **REQ-OBS-02 – Aggregation levels.** Statistics shall be computed for the whole detector and per-chip.
3. Must **REQ-OBS-03 – Hit-rate UI cadence.** While Running or Previewing, the UI shall update hit-rate at least once per second.
4. Should **REQ-OBS-04 – Per-stage throughput (opt-in).** When observability is enabled, each pipeline stage shall report throughput in bytes/s and queue depths; this view is off by default.
5. Must **REQ-OBS-05 – Latency reporting.** The system shall compute ingest→present latency (and ingest→disk latency for Acquire) and update a 1-s rolling mean every 0.5 s.
6. Must **REQ-OBS-06 – Observability overhead bound.** With observability enabled, median ingest-¿present latency shall not degrade by more than +10 ms vs. baseline.
7. Could **REQ-OBS-07 – Scan–detector sync metric.** Provide a sync metric estimating jitter between scan-box timing and detector hit timestamps, e.g., $J = \mathrm{abs}(t_{\mathrm{hit@40MHz}} - t_{\mathrm{scan}})$, with histogram and summary stats.
8. Must **REQ-OBS-08 – Dropped-data visibility.** Dropped/lost/mis-mapped event counters shall be tracked per second and surfaced in UI and logs, with clear warnings when non-zero.

3.1.10 UI Shell

This section describes the UI counterpart of the feature only. Every UI component is backed by a non-UI component and its business logic is tested away from the UI in the form of unit/integration/end-to-end testing.

1. Must **REQ-UI-01 – Standalone launch.** The application shall launch as a standalone process when the user double-clicks the 4D-STEM UI icon or selects it from the native OS menu.
2. Must **REQ-UI-02 – Restricted detector controls.** The UI shall display only the detector controls listed in whitelist (still need to write the whitelist).

   Acceptance: Across all screens and states, only whitelisted controls appear; no navigation path exposes non-whitelisted controls.
3. Must **REQ-UI-03 – Derived detector settings.** Where possible, detector settings shall be derived automatically from scan parameters without additional user interaction. If the setting can be abstracted from the user, then it should be.
4. Must **REQ-UI-04 – Scan engine Configuration.** The UI shall present a control window to configure the available scan engine. The scan engine must be supported by software.
5. Must **REQ-UI-05 – Scan geometry controls.** The UI shall present controls for geometry including: scan height and width (in probe positions), prescan x/y, always visible.
6. Must **REQ-UI-06 – Dwell control.** The UI shall present a dwell-time control, always visible.
7. Must **REQ-UI-07 – Z-frames control.** The UI shall present a control for the number of frames Z (frame = one full scan of $H \times W$), always visible.
8. Must **REQ-UI-08 – Start (Preview).** The UI shall provide a Start (Search/Preview) control.
9. Must **REQ-UI-09 – Start (Acquire).** The UI shall provide a Start (Acquire) control.
10. Must **REQ-UI-10 – Output Location.** The UI presents a widget to collect output location. The widget accepts text input and selection from the native filesystem widget
11. Must **REQ-UI-11 – Output format.** A dropdown containing supported output formats is presented and always visible.
12. Must **REQ-UI-12 – Workspaces.** Users shall be able to create/open a workspace; a workspace can contain multiple measurements.
13. Must **REQ-UI-13 – Measurement metadata.** Each measurement shall have metadata persisted as JSON (schema defined in Data Formats), accessible via the UI but hidden by default. All scan and detector parameters are included in the measurement metadata plus comments written by users themselves.
14. Must **REQ-UI-14 – Stop button.** The UI shall provide a Stop control that is enabled only while Preview or Acquire is running.
15. Must **REQ-UI-15 – Stop semantics.** When the user presses *Stop*, the system shall finish the current probe position and then stop. A preference 'Stop at frame boundary' shall, when enabled, finish the current frame before stopping.
16. Must **REQ-UI-16 – ROIs.** The user shall be able to create, update/move and remove rectangular, circular or annular ROIs by click-and-drag on the viewport.
17. Must **REQ-UI-17 – ROI to scan geometry.** The system shall translate a rectangular ROI into an equivalent scan geometry. ROIs snap to the pixel grid so values derived are integers.
18. Must **REQ-UI-18 – ROI for diffraction views.** The system shall allow using ROI selections to build diffraction images (e.g. averaged CBED of the selection).
19. Could **REQ-UI-19 – ROI expanded view.** A ROI shall be expandable via invoking the 'right click ¿ Expand ROI' function. The system produces a full size, zoomed in image from the selection and optionally enables updating the scan geometry to reflect the new sub-region
20. Could **REQ-UI-20 – Dockable controls.** Control panels shall be dockable/undockable and toggleable via the View menu. A default layout is exposed for return to factory settings.
21. Must **REQ-UI-21 – Persistence.** UI settings (layout, last workspace, panel visibility, scan control values) shall persist across sessions and at the granularity of a workspace. Detector specific controls persist per workstation, rather than per workspace.
22. Must **REQ-UI-22 – Connectivity loss handling.** On loss of connectivity to Serval, scan engine, or detector, the UI shall display status and attempt reconnect every 1 s until success.

23. Must **REQ-UI-23 – Startup without connectivity.** If no connectivity is available at startup, the UI shall present a non-blocking status and allow offline configuration. Any setting or control relating to the unconnected device or server shall be blocked when not connected.

24. Must **REQ-UI-24 – Logging.** The UI shall write a log of important events to disk and to console; log level is controlled by environment variable `LOG_LEVEL`. Log location is user-configurable under `File > Preferences` but defaults to within the workspace configuration.

25. Must **REQ-UI-25 – Disk space guard (preflight + warning).** On *Start (Acquire)*, the system shall evaluate free space on the selected target volume. If free space is at or below the near-full threshold (default 5% of the volume), the system shall present a low-space warning before starting. If the volume reports 0 writable bytes free, the acquisition shall not start.

26. Must **REQ-UI-26 – Detector info and health available.** UI shall display from view ¿ detector health a UI panel describing the detectors information from serval

3.2 Non-Functional Requirements

Data Integrity: Data measured using our device is sacred. We should do our utmost to ensure we collect all data requested by the user as accurately as possible. By default, the data shall not be modified in any way storage. This philosophy is shared with serval; output should be exactly as recorded, regardless of known problems in the hardware that need correction, including the tramlane effect and the VCC correction. Nevertheless, it is convenient for users to have these things automatically corrected so that they do not need to do it themselves. We can support an opt-in switch that toggles corrections in the raw data.

Performance: There are some performance-critical operations proposed in this software. Namely, the live processing needs to be able to keep up with the data acquisition and for adequate user feedback for the microscopist we need to minimize latency. Offline processing does not need the same performance requirement and just needs to run 'as fast as reasonable'.

Scalability: In the context of processing data from timepix detectors, scalability means the ability to handle more data at once. The processing requirements scale with the detector type and layout. The tpx3 single chip can read out at a rate of 40Mbits/s while the 'turbo' edition can operate at 80Mbits/s. There are single, quad and 2x4 variations in the layout of detectors. Performance requirements scale linearly with the addition of every chip. So respectively, the 1, 4, and 8 chip detectors can read out a maximum of 80, 320 and 640Mbits/s and half this for the non-turbo versions. A single tpx4 device can read out a maximum of 160Gbits/s, which presents a challenge on an entirely new scale for data processing. Yet despite the engineering challenges, it is imperative that we build for the future generation of timepix as well as focusing on the now.

3.3 Legal and Regulatory Requirements

Licenses. What can we do? What can't we do?