Towards the Camera Calibration Data Model - Proposals

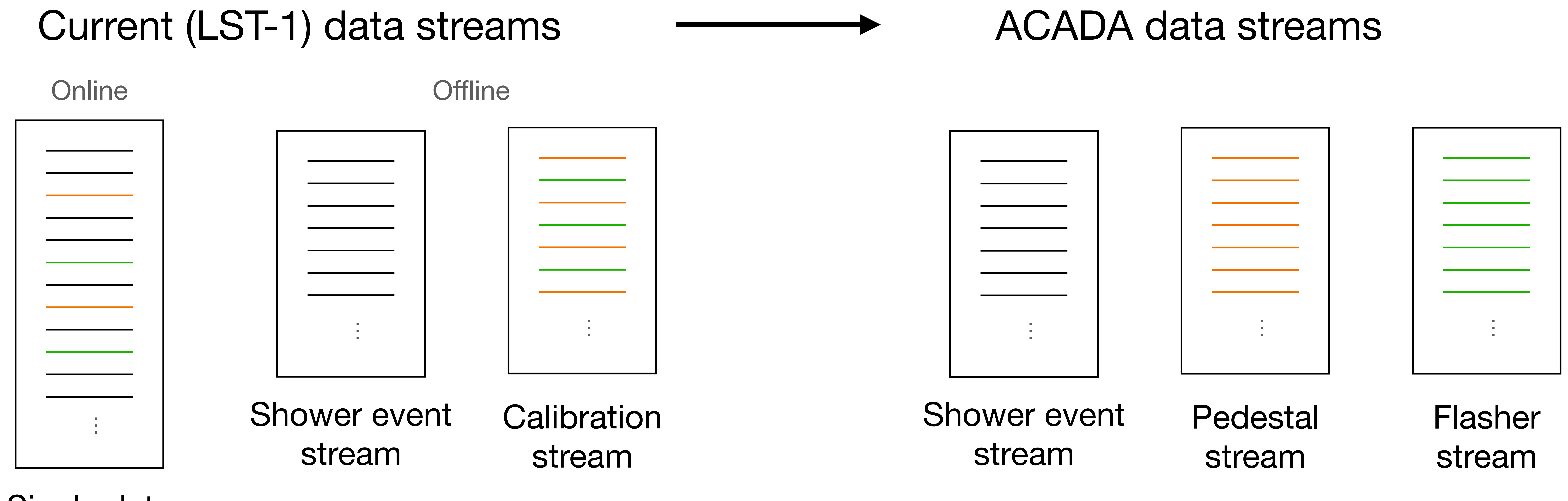
Tjark Miener (tjark.miener@unige.ch) on behalf of the CalibPipe team

DataPipe Hackathon 24.10.2023 - 27.20.2023

Outline

- Data streams
- Data Model Proposals
  - Proposal A: current `ctapipe/lstchain` data structure
  - Proposal B & C: CTAO *DL0* data structure
- Static vs dynamic data storing
- Timestamps vs event IDs
- Discussion

Data streams



- ACADA will provide the DPPS with distinct data streams for the calibration data
- Pedestal and flat-field parameters should be processed independently and stored in their corresponding containers
- Camera calibration coefficients should be then calculated from those internal files and store permanently in a data model to interface the other DPPS pipelines

Current *ctapipe/lstchain* data structure

- Proposal A is designed to match the current `ctapipe/lstchain` data structure
- In place solution that is tested on a daily basis (mainly for the LST-1 prototype in monoscopic mode). Current calibrated *R1*-waveforms do not follow the *DL0* Data Model which will be provided to the DPPS by ACADA.

Proposal A.1: Current *ctapipe/lstchain* data structure (static storing)

Scheme based on timestamps

Table A.1 – Flat-field parameters proposal A.1.

Name	Type	Description
sample_time	float64	Time (in s) associated to the considered sample.
sample_time_min	float64	Minimum time (in s) of the considered sample.
sample_time_max	float64	Maximum time (in s) of the considered sample.
num_samples	uint64	Number of events used for the calculation of flat-field parameters.
signal_charge [num_channels, num_pixels]	Measurement::BasicType	Measurement of the signal charge over the considered sample in all channels and pixels.
signal_time [num_channels, num_pixels]	Measurement::BasicType	Measurement of the signal time over the considered sample in all channels and pixels.
relative_gain [num_channels, num_pixels]	Measurement::BasicType	Measurement of the relative gain over the considered sample in all channels and pixels.
relative_time [num_channels, num_pixels]	Measurement::BasicType	Measurement of the relative time over the considered sample in all channels and pixels.
outliers_charge [num_channels, num_pixels]	Measurement::BasicType	Measurement of the charge outliers the considered sample in all channels and pixels.
outliers_time [num_channels, num_pixels]	Measurement::BasicType	Measurement of the time outliers over the considered sample in all channels and pixels.
failing_pixels [num_channels, num_pixels]	bool	True if the algorithm failed to produce valid flat-field parameters in all channels and pixels.

Proposal A.2: Current *ctapipe/lstchain* data structure (static storing)

Scheme based on event IDs

Table A.2 – Flat-field parameters proposal A.2.

Name	Type	Description
event_start	uint64	ID of the first event in the considered sample.
event_stop	uint64	ID of the last event in the considered sample.
num_samples	uint64	Number of events used for the calculation of flat-field parameters.
signal_charge [num_channels, num_pixels]	Measurement::BasicType	Measurement of the signal charge over the considered sample in all channels and pixels.
signal_time [num_channels, num_pixels]	Measurement::BasicType	Measurement of the signal time over the considered sample in all channels and pixels.
relative_gain [num_channels, num_pixels]	Measurement::BasicType	Measurement of the relative gain over the considered sample in all channels and pixels.
relative_time [num_channels, num_pixels]	Measurement::BasicType	Measurement of the relative time over the considered sample in all channels and pixels.
outliers_charge [num_channels, num_pixels]	Measurement::BasicType	Measurement of the charge outliers the considered sample in all channels and pixels.
outliers_time [num_channels, num_pixels]	Measurement::BasicType	Measurement of the time outliers over the considered sample in all channels and pixels.
failing_pixels [num_channels, num_pixels]	bool	True if the algorithm failed to produce valid flat-field parameters in all channels and pixels.

Proposal A.2: Current *ctapipe/lstchain* data structure (static storing)

Scheme based

Measurement::BasicType		
num_samples	uint64	Number of samples used in this measurement.
sample_sum	BasicType	Total summation of the entries of the sample.
sample_squared_sum	float64	Squared summation of the entries of the sample.
min	BasicType	Minimum value from the sample.
max	BasicType	Maximum value from the sample.
std	float64	Standard deviation from the sample.
mean	float64	Arithmetic mean calculated from the sample.

Name			
event_st			
event_st			
num_sa			-field param-
signal_charge	[num_channels, num_pixels]	Measurement::BasicType	Measurement of the signal charge over the considered sample in all channels and pixels.
signal_time	[num_channels, num_pixels]	Measurement::BasicType	Measurement of the signal time over the considered sample in all channels and pixels.
relative_gain	[num_channels, num_pixels]	Measurement::BasicType	Measurement of the relative gain over the considered sample in all channels and pixels.
relative_time	[num_channels, num_pixels]	Measurement::BasicType	Measurement of the relative time over the considered sample in all channels and pixels.
outliers_charge	[num_channels, num_pixels]	Measurement::BasicType	Measurement of the charge outliers the considered sample in all channels and pixels.
outliers_time	[num_channels, num_pixels]	Measurement::BasicType	Measurement of the time outliers over the considered sample in all channels and pixels.
failing_pixels	[num_channels, num_pixels]	bool	True if the algorithm failed to produce valid flat-field parameters in all channels and pixels.

TelescopeEvent and PixelWaveform structure of *DL0* Data Model

Telescope Event			
event_id	uint64	The event ID assigned by the subarray trigger.	•
tel_id	uint16	The ID of the Telescope whose Camera sent the event.	•
event_type	uint8	The type of event, see definition in Sec. A.1.4.	
event_time	HiResTimeStamp	High-precision timestamp assigned by the Cherenkov Camera.	
pixel_status	uint8 [num_pixels]	See definition in Sec. A.1.5.	
first_cell_id	uint16 [N]	The cell id of the first cell in the ring-sampler that is read out (*optional*, N is 1 for SST, and 2120 for LST (265 modules * 8 chips/module)).	
num_channels	uint8	The number of channels in this event. The only possible values are 1 or 2.	
calibration_monitoring_id	uint64	ID of the CalibrationMonitoringSet containing the applied pre-calibration parameters.	
PixelWaveform			
pixel_id	uint64	ID of the pixel.	•
waveform	uint16 [num_channels ~~num_pixels~~, num_samples]	R1-calibrated time-sampled image series in photoelectrons. *Notes:* (i) num_samples will be num_samples_nominal for all cases except when event_type==33, in which case num_samples_long will be used (ii) It is also possible to store images in DCs when the associated pre-calibration coefficients are set to 1, (iii), data volume reduction is already applied by ACADA to these data.	
pedestal_intensity	float32 [num_pixels]	Pedestal (baseline) intensity in each pixel in DC, and potentially additional information from the baseline. (*optional*).	

TelescopeEvent and PixelWaveform structure of *DL0* Data Model

- Proposals B and C are designed to match the *DL0* data structure

- Even though the CamCalib belongs to the *DL1* production, we might want to align Data Model for the camera calibration coefficients with the *DL0* data structure since their application is the first task perform by `DataPipe`.

- *DL0* data volume reduction results in compress data and therefore only a specific region of interest needs to be calibrated

Proposal B: Designed to match *DL0* data structure (static storing)

Table A.3 – Flat-field parameters proposal B.

Name	Type	Description
pixel_id	uint64	ID of the pixel. (TBD: this could be stored in the metadata)
event_start	uint64	ID of the first event in the considered sample.
event_stop	uint64	ID of the last event in the considered sample.
num_samples	uint64	Number of events used for the calculation of flat-field parameters.
signal_charge [num_channels]	Measurement::BasicType	Measurement of the signal charge over the considered sample in pixel_id.
signal_time [num_channels]	Measurement::BasicType	Measurement of the signal time over the considered sample in pixel_id.
relative_gain [num_channels]	Measurement::BasicType	Measurement of the relative gain over the considered sample in pixel_id.
relative_time [num_channels]	Measurement::BasicType	Measurement of the relative time over the considered sample in pixel_id.
outliers_charge [num_channels]	Measurement::BasicType	Measurement of the charge outliers over the considered sample in pixel_id.
outliers_time [num_channels]	Measurement::BasicType	Measurement of the time outliers over the considered sample in pixel_id.
failing_pixels [num_channels]	bool	True if the algorithm failed to produce valid flat-field parameters in pixel_id.

Static vs dynamic: Store coefficients as onchange events

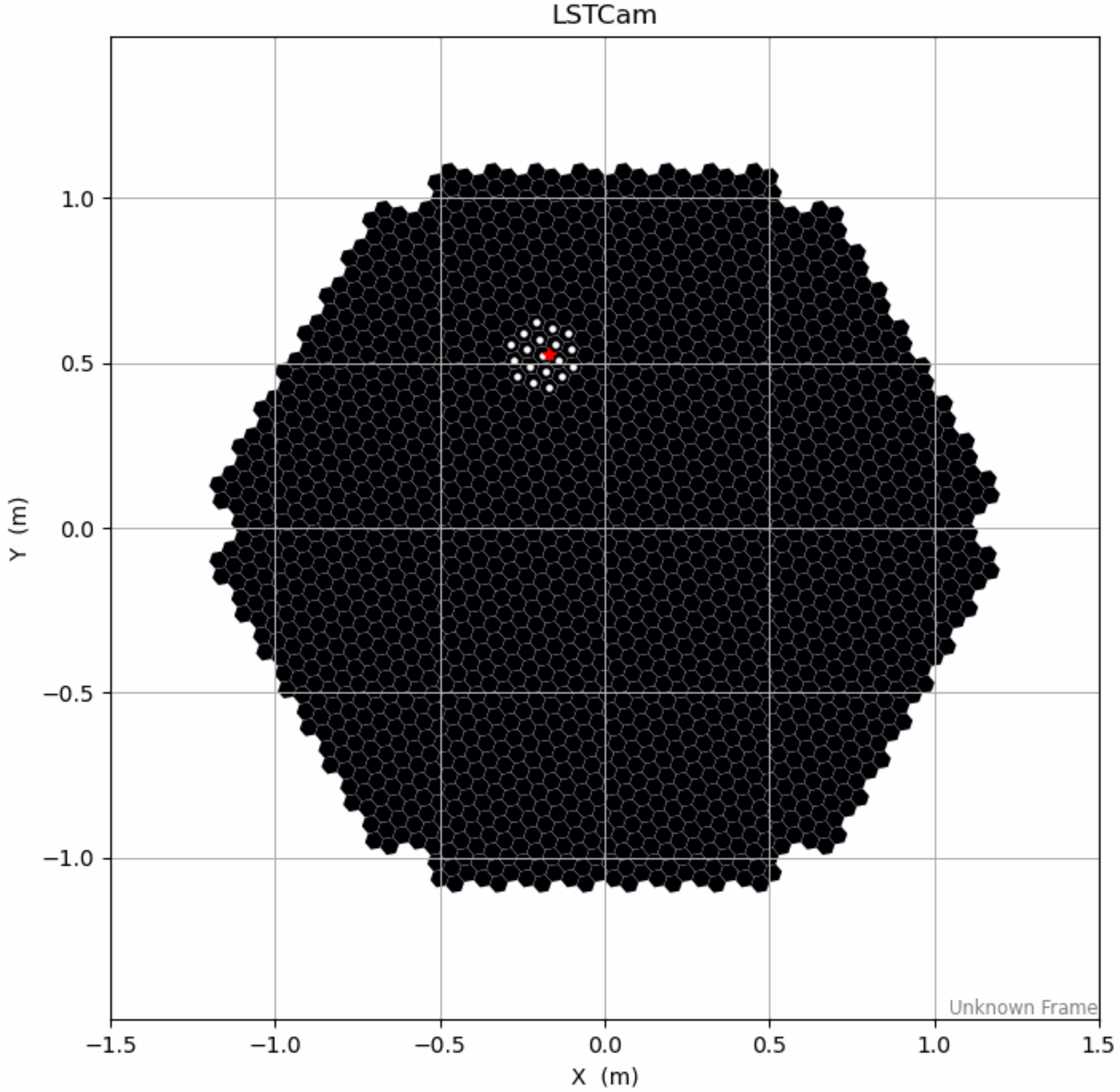
- Proposals A and B are designed for a static data storing. Pedestal/ flat-field parameters and camera calibration coefficients are stored every iteration.
- Proposal C is designed for a dynamic data storing that is only saving parameters and coefficients to disk when they change.
- Pedestal and flat-field parameters are expected to be relatively stable in a given uncertainty at standard data taken. Stars in the field of view are suppose to move rather slow in comparison to the calculation of the parameters.
- Dynamic data storing would also allow a dynamic sample window for the calculation of the parameters.

Proposal C: Designed to match *DL0* data structure (dynamic storing)

Table A.4 – Flat-field parameters proposal C.

Name	Type	Description
pixel_id	uint64	ID of the pixel. (TBD: this could be stored in the metadata)
event_start	uint64	ID of the first event in the considered sample.
signal_charge [num_channels]	Measurement::BasicType	Measurement of the signal charge over the considered sample in pixel_id.
signal_time [num_channels]	Measurement::BasicType	Measurement of the signal time over the considered sample in pixel_id.
relative_gain [num_channels]	Measurement::BasicType	Measurement of the relative gain over the considered sample in pixel_id.
relative_time [num_channels]	Measurement::BasicType	Measurement of the relative time over the considered sample in pixel_id.
outliers_charge [num_channels]	Measurement::BasicType	Measurement of the charge outliers over the considered sample in pixel_id.
outliers_time [num_channels]	Measurement::BasicType	Measurement of the time outliers over the considered sample in pixel_id.
failing_pixels [num_channels]	bool	True if the algorithm failed to produce valid flat-field parameters in pixel_id.

Discussion



Low precision timestamps vs event IDs

- Timestamps are common identifiers which allows to select a matching sampling window for the pedestal and flasher events.

- With event IDs (and the obs ID) one could reconstruct the timestamps. However, the sampling window for a data structure based on event IDs would not match to 100%.

Current scheme based on timestamps

Name	Type	Description
sample_time	float64	Time (in s) associated to the considered sample.
sample_time_min	float64	Minimum time (in s) of the considered sample.
sample_time_max	float64	Maximum time (in s) of the considered sample.
num_samples	uint64	Number of events used for the calculation of flat-field parameters.
signal_charge [num_channels, num_pixels]	Measurement::BasicType	Measurement of the signal charge over the considered sample in all channels and pixels

Scheme based on event IDs

Name	Type	Description
event_start	uint64	ID of the first event in the considered sample.
event_stop	uint64	ID of the last event in the considered sample.
num_samples	uint64	Number of events used for the calculation of flat-field parameters.
signal_charge [num_channels, num_pixels]	Measurement::BasicType	Measurement of the signal charge over the considered sample in all channels and pixels

Discussion

Table 1.1 – Discussion on the camera calibration Data Model.

	Advantages	Disadvantages
Proposal A		
This proposal is designed to match the current `ctapipe` data structure (static storing).		
	• allow to store the relevant information of the whole camera in one compact data structure/table	• waveform data of *DL0/Event/Telescope* has a pixel-wise data structure from the PixelWaveform container
	• camera calibration coefficients are calculated over a static time interval assuming a constant rate of data taken	• camera calibration coefficients of each pixels have to be read even though only a fraction of pixels may be needed for the compressed `DL0` data
		• stable camera calibration coefficients are stored every iteration
Proposal B		
This proposal is designed to match the DL0/Event/Telescope data structure (static storing).		
	• camera calibration coefficients follows the pixel-wise data structure from the PixelWaveform container	• `DataPipe` needs to assemble the shower images from the pixel-wise calibration data
	• camera calibration coefficients are calculated and stored over a static time interval assuming a constant rate of data taken	• stable camera calibration coefficients are stored every iteration
Proposal C		
This proposal is designed to match the DL0/Event/Telescope data structure (dynamic storing).		
	• camera calibration coefficients follows the pixel-wise data structure from the PixelWaveform container	• `DataPipe` needs to assemble the shower images from the pixel-wise calibration data
	• allow to store pixel-wise camera calibration coefficients dynamically when they change	• stored camera calibration coefficients have different time intervals depending on the pixels
	• sliding window (number of samples) used for the calculation can be dynamic	

Main discussion points

- Should be use timestamps or event IDs for defining the sample window?
- Would a dynamic sample window for the calculation of the coefficients be useful and practical?
- At which stage we should assemble the shower images from the pixel-wise *DL0* and calibration data?
- Different proposal better suited for Cat-B or Cat-C calibration calculation?

Merci pour votre attention!