



The Instrumental Edge: Real-Time and AI-Ready Discovery

Adam Thompson – Head of Product – HPC and AI at the Edge

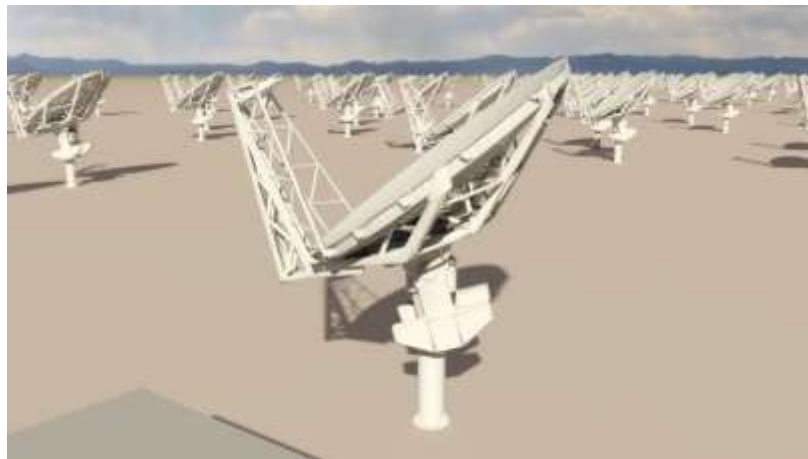
adamt@nvidia.com

Next Generation Instruments are Producing Overwhelming Amounts of Data

Complexity of Experiments is Booming and Human Insight is Now the Bottleneck

Radio Astronomy

ngVLA – 244 Dishes
100 Petabytes per Year



SKA – 200 Dishes
1 Exabyte per Year

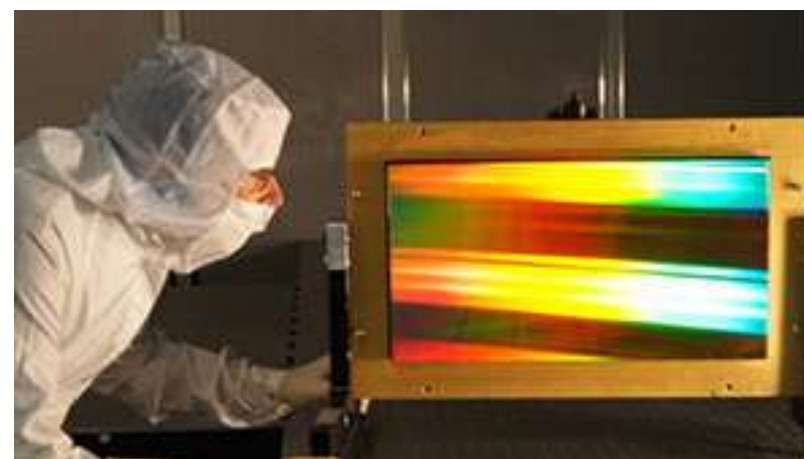


Particle Physics

High Luminosity LHC
100x Data than Higgs Boson Discovery



Advanced Laser Systems
100x Increase in Repetition Rate



Light Sources

APS-U – >60 beamlines
100-200 Petabytes per Year

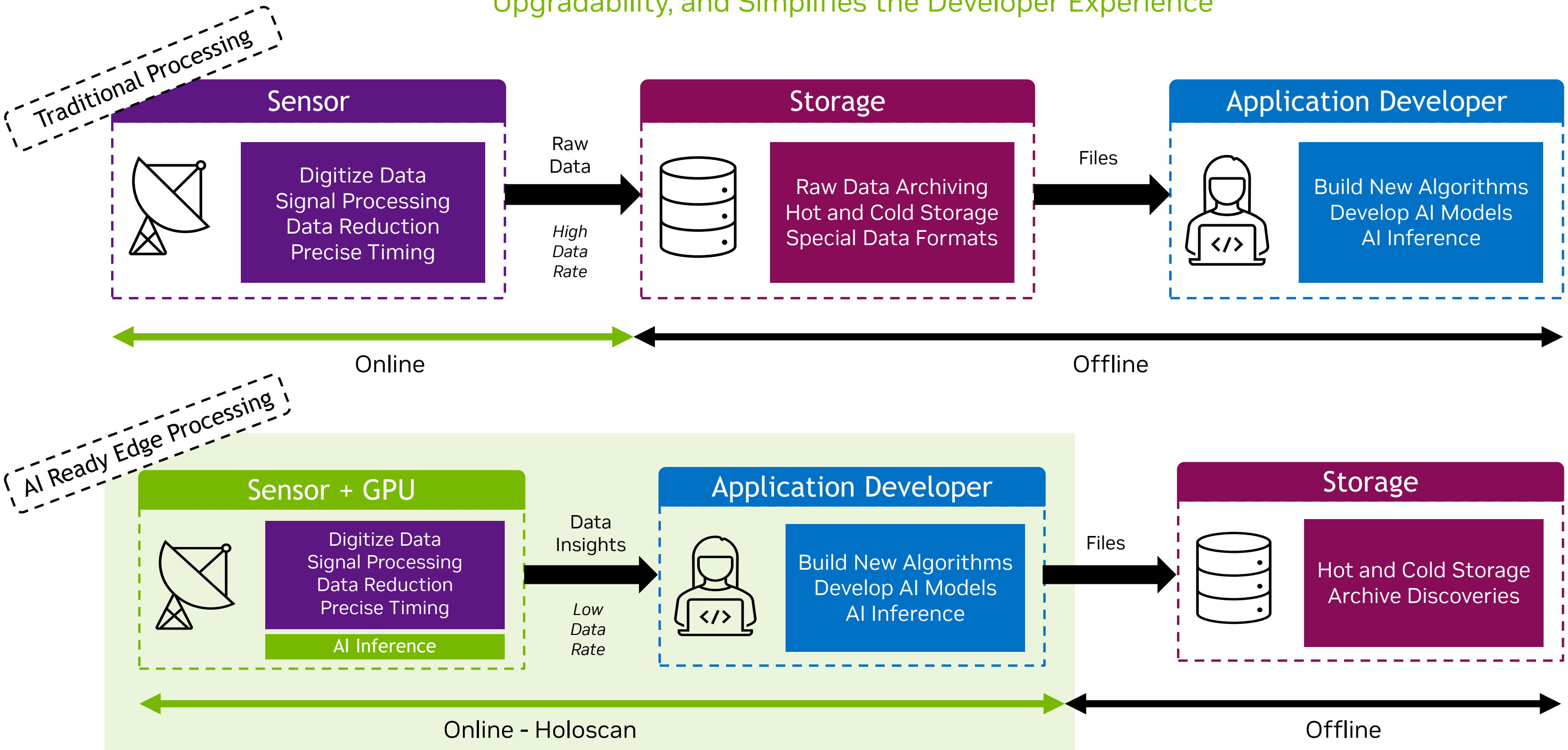


Free Electron Laser LCLS-II
1 MHz Rep-Rate Upgrade



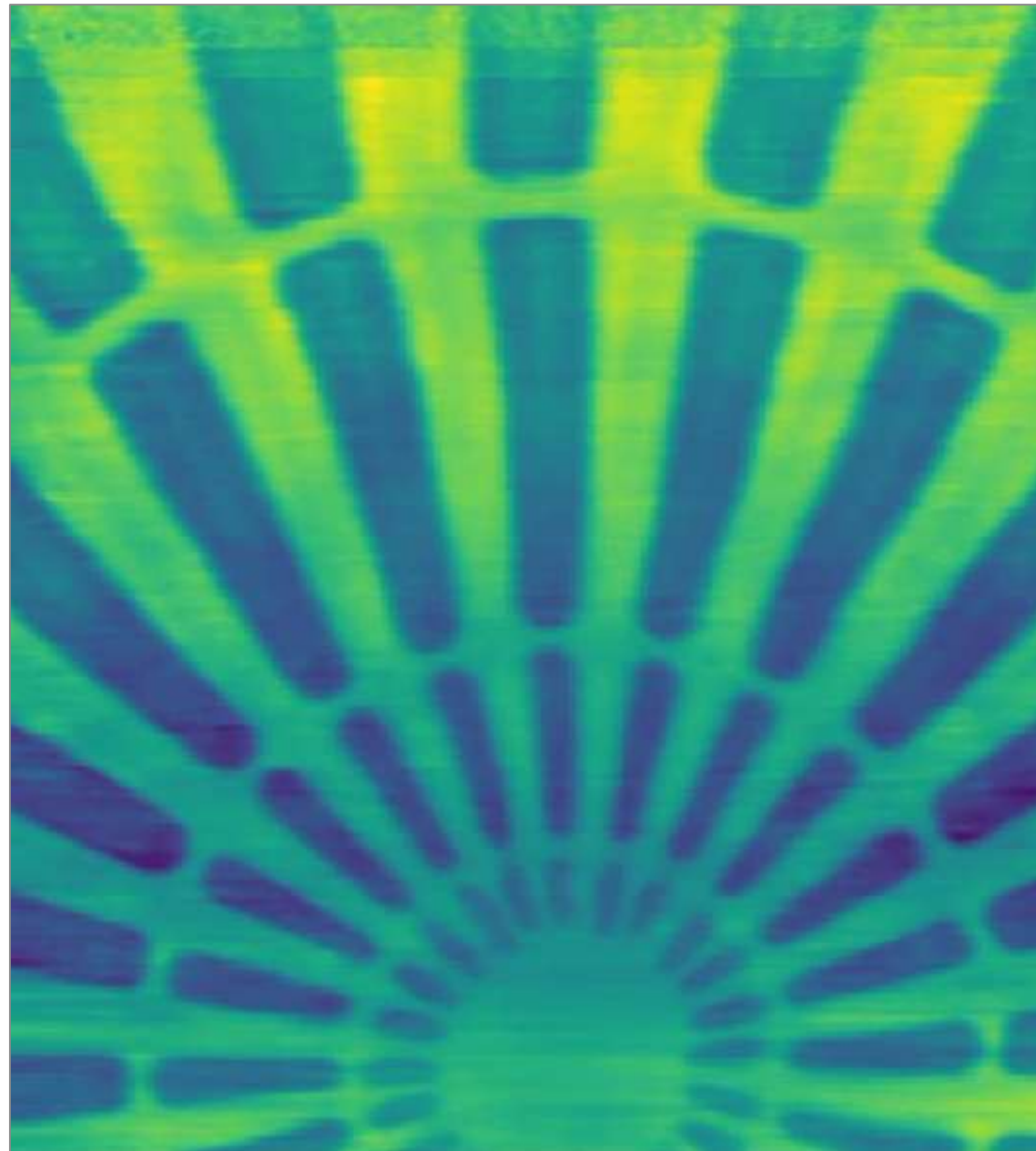
Resolving the Data Deluge with HPC and AI at the Edge

Online AI Processing Enables Autonomous Experiments, Reduces Infrastructure Costs, Improves Upgradability, and Simplifies the Developer Experience



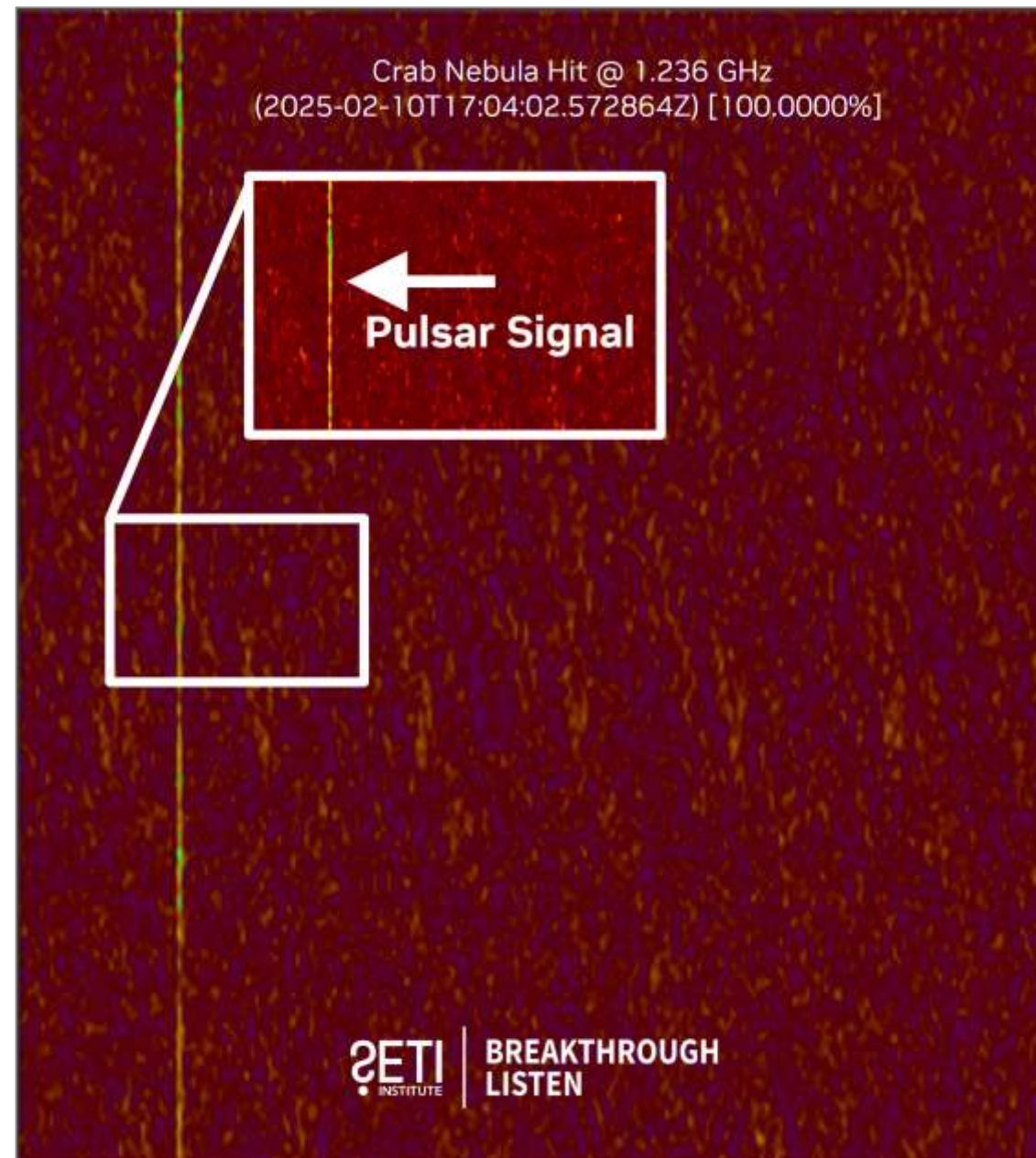
Real Time Data Analysis at the Edge

AI, Combined With Streaming Data and Unprecedented Rates, Welcomes New Scientific Discoveries



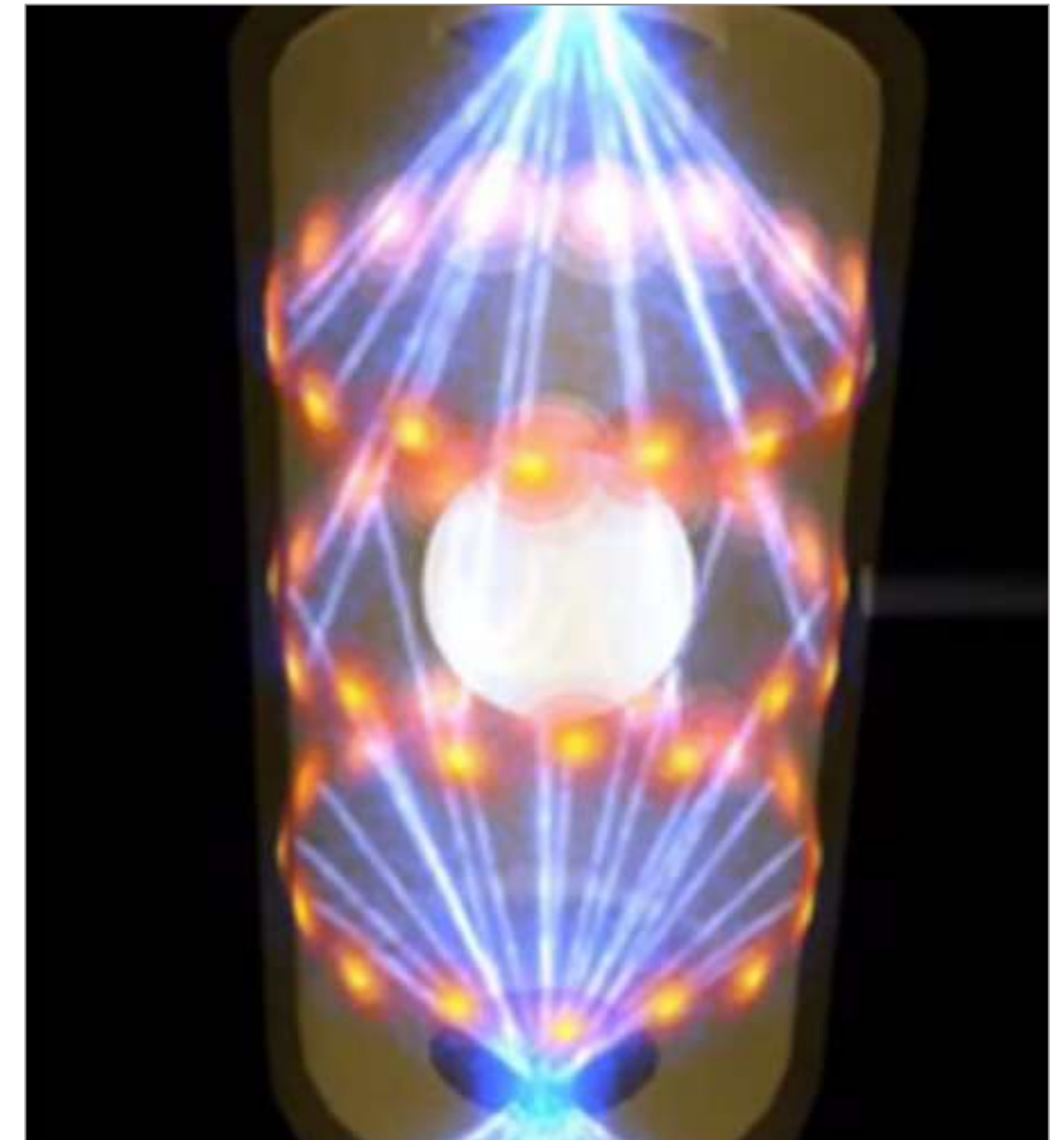
Nanoscale Imaging of Materials

Real Time Ptychography with NSLS-II and DECTRIS



AI Search for Pulsars and Technosignatures

AI as a Detector with the SETI Institute and the Allen Telescope Array

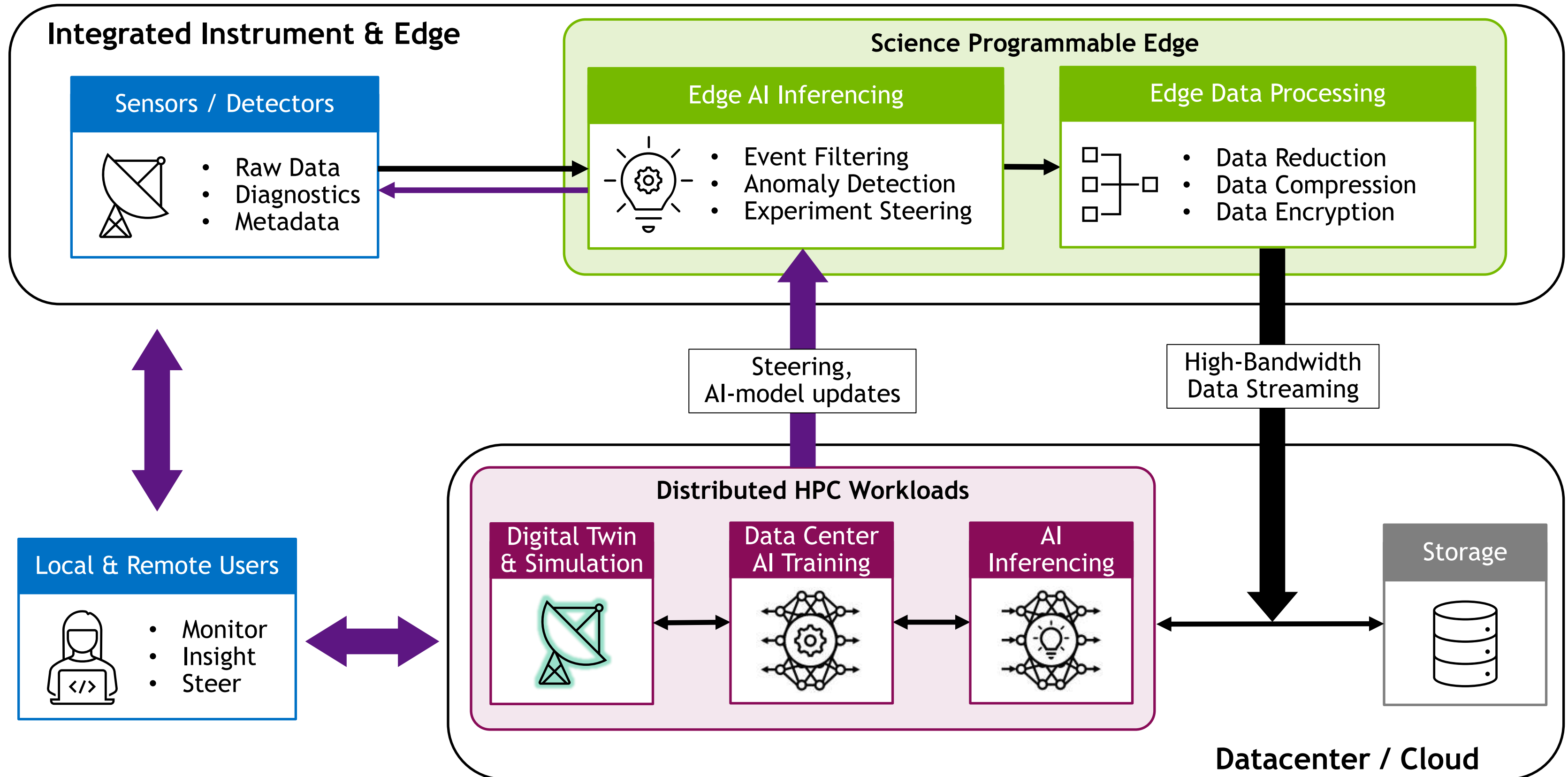


Self Driving Experiments

AI Guided Fusion Research with Lawrence Livermore National Laboratory

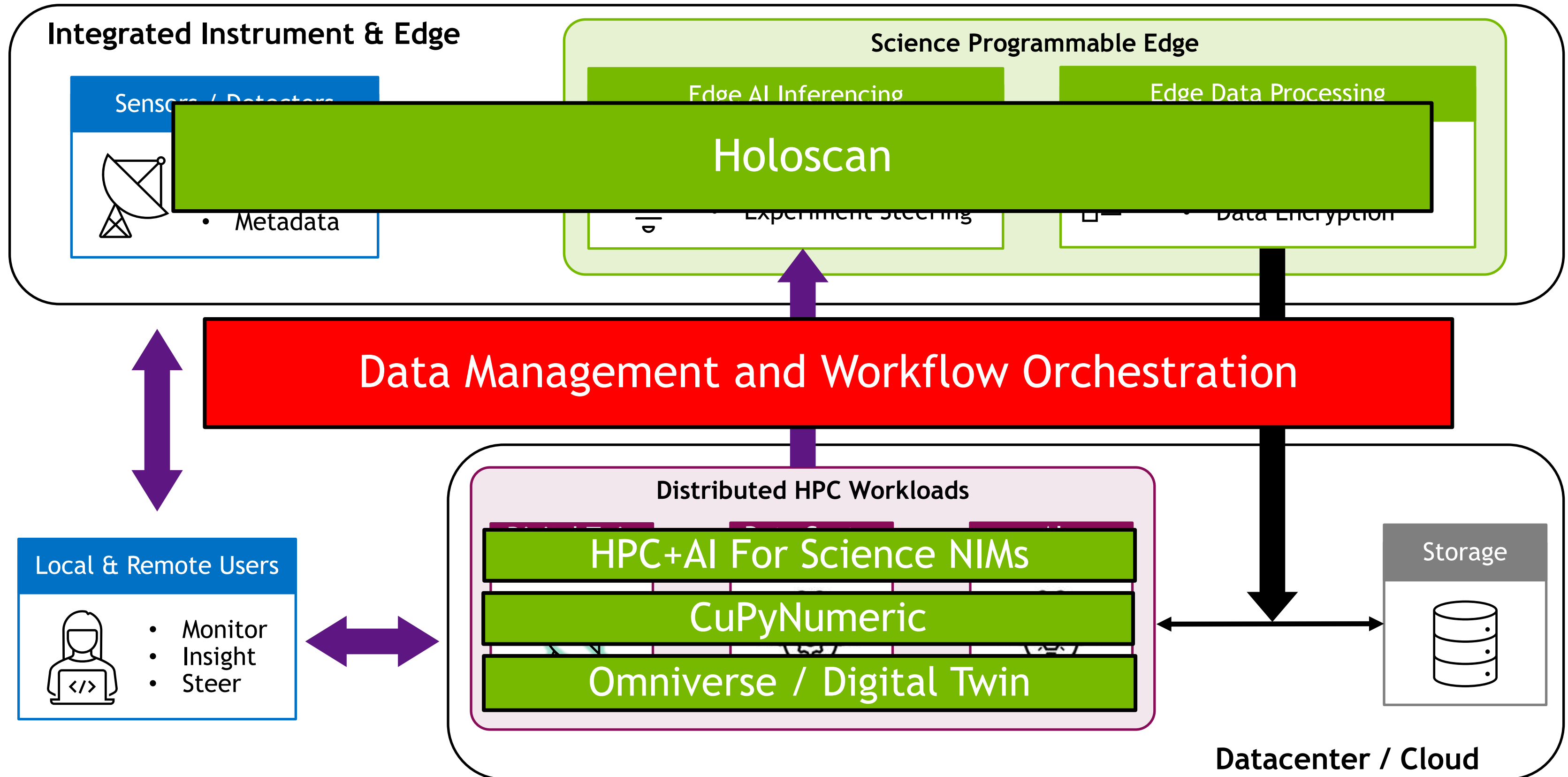
Future Instruments Will Combine Edge and Distributed HPC Workloads

A Vision of Towards Integrated Ecosystem for Self-Driving Experiments and Laboratories



Future Instruments Will Combine Edge and Distributed HPC Workloads

A Vision of Towards Integrated Ecosystem for Self-Driving Experiments and Laboratories





Holoscan

**AI-Enabled, Real-Time Sensor
Processing Platform**



GitHub

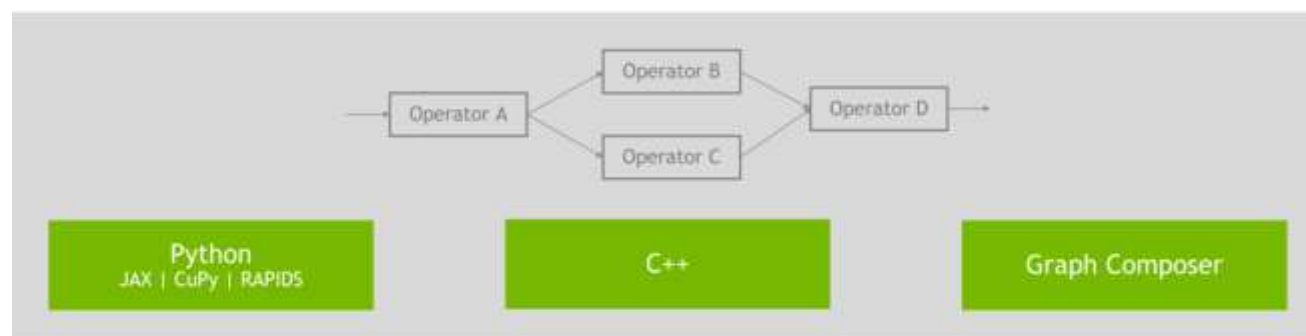
NVIDIA Holoscan

SDK for Building AI-Enabled Sensor Processing Applications



HoloHub
Operators | Reference Apps

Model Zoo
NGC | MONAI



Operators

IO AI Inference Viz Custom

DPDK Rivermax TensorRT Triton Vulkan

CUDA-X



Features

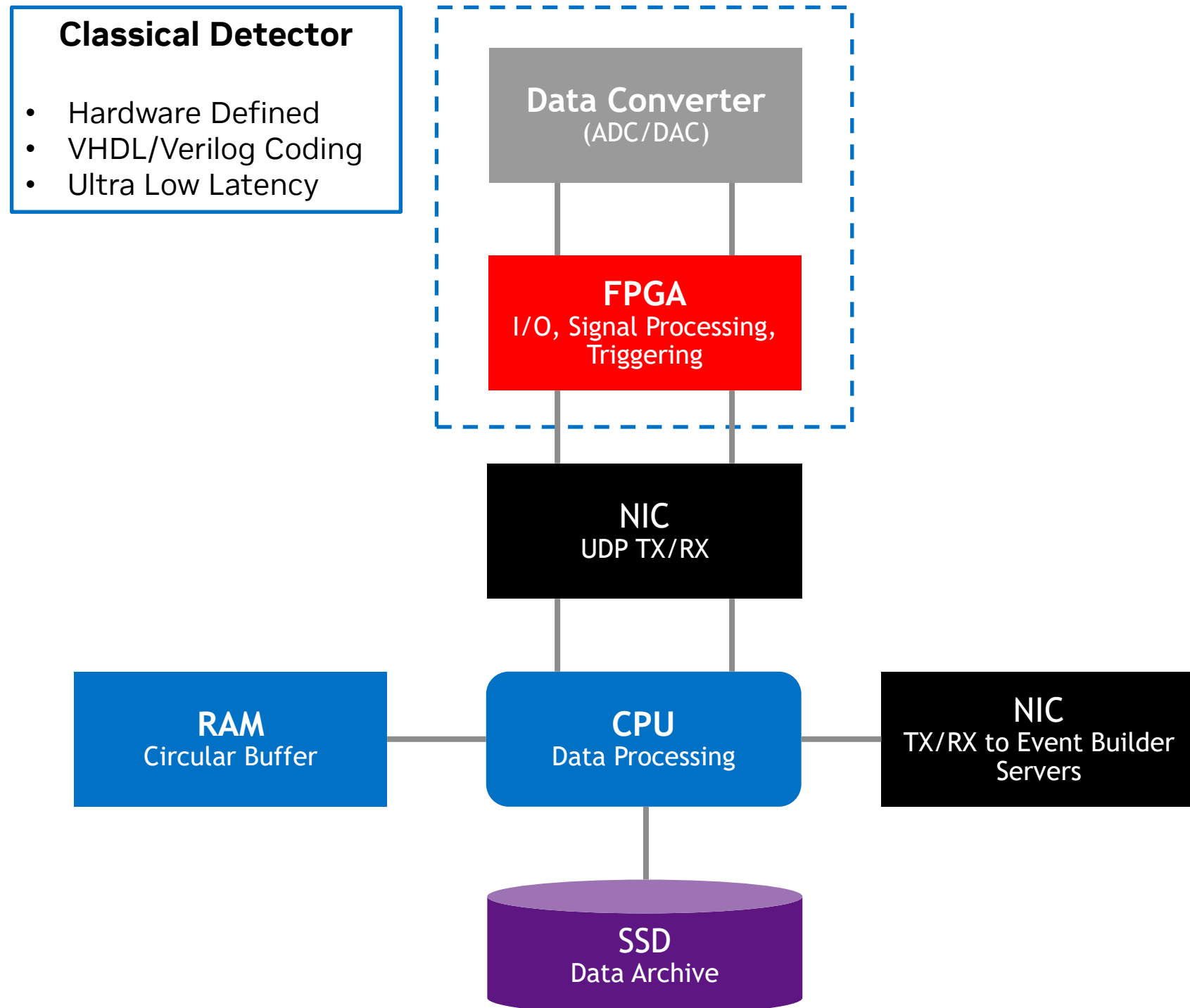
- C++ and Python APIs for **domain agnostic** sensor data processing workflows
- Multi-Node and Multi-GPU support with advanced pipeline scheduling options and network-aware data movement
- AI Inference with pluggable backends such as ONNX, Torchscript, and TensorRT
- Scalable from Jetson Orin Nano (ARM + GPU) to Grace Hopper
- Apache 2 Licensed and Available on [GitHub](#)

Benefits

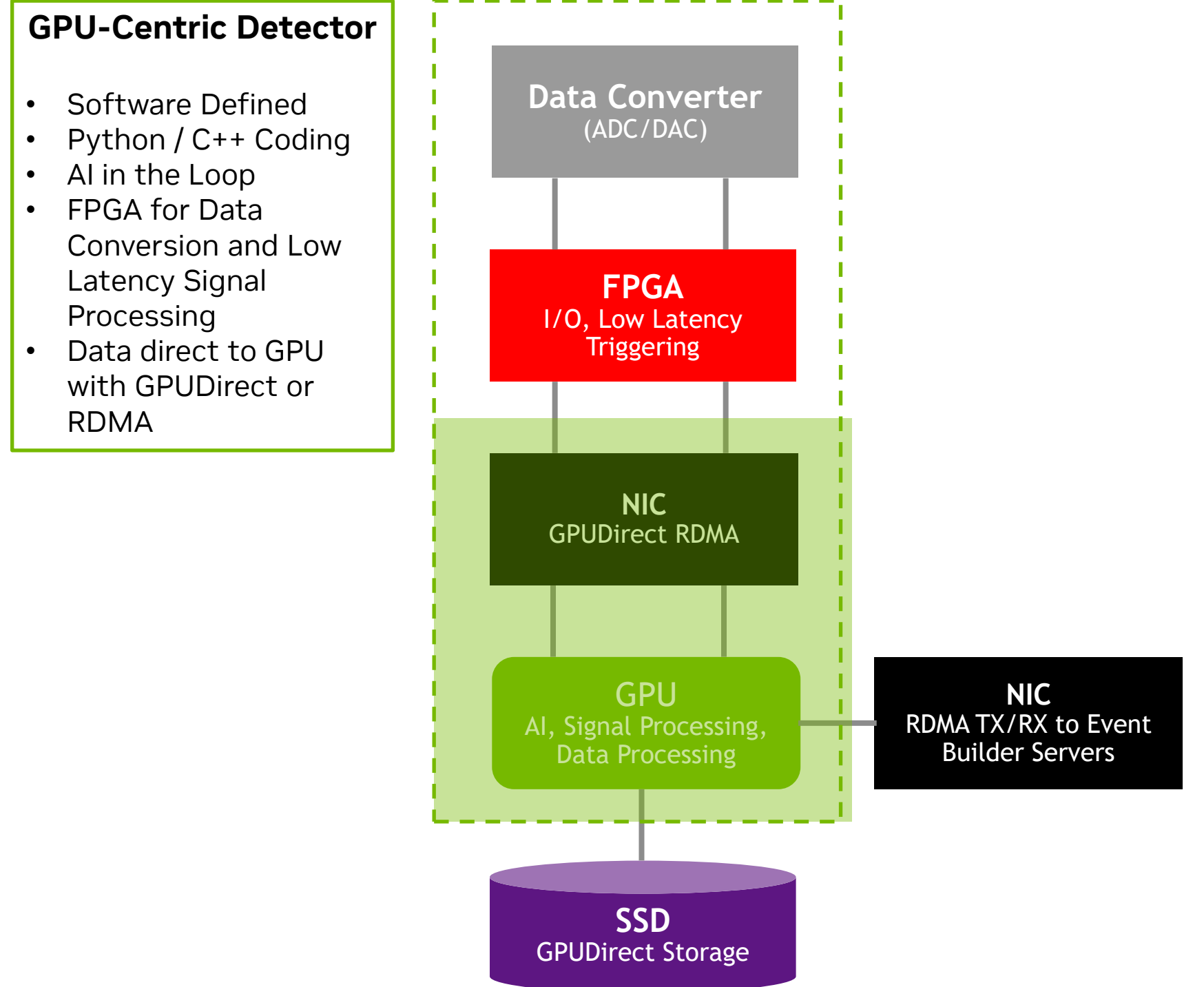
- Simplifies sensor I/O to GPU
- Simplifies the performant deployment of an AI model in a streaming pipeline
- Provides customizable, reusable, and flexible components to build and deploy GPU-accelerated algorithms
- Scale workloads with Holoscan's distributed computing features
- Deploy to the Cloud with Holoscan App Packager and Kubernetes

AI Enabled and Science Programmable Data Acquisition (DAQ)

Traditional DAQ Architecture



AI Ready DAQ Architecture



Holoscan Advanced Network Operator

Simplification of Moving Data To and From GPU at Line Rate

Advanced Network Operator

Focus on **Performance**: Any Ethernet Packet (UDP, VITA-49, etc) to and from GPU at Line Rate

Bypasses Linux kernel for access directly to NIC DMA buffers

Requires a Mellanox/NVIDIA Network Interface Card (NIC)

Zero-copy interface from NIC into user buffers or directly to GPU

Supports multiple backends (DOCA DPDK, Rivermax, GPUNetIO, RoCEv2) with YAML configuration

Python bindings are in progress

Learn more about the [ANO on HoloHub](#)



YAML Configuration - Header Data Split

```
memory_regions:
- name: "Data_RX_CPU"
  kind: "huge"
  affinity: 0
  num_bufs: 51200
  buf_size: 64
- name: "Data_RX_GPU"
  kind: "device"
  affinity: 0
  num_bufs: 51200
  buf_size: 1000

interfaces:
- name: "rx_port"
  address: 00:05.1      # The BUS address of the interface doing Rx
  rx:
    flow_isolation: true
    queues:
    - name: "rq_q_0"
      id: 0
      cpu_core: 9
      batch_size: 10240
      memory_regions:
      - "Data_RX_GPU"
      - "Data_RX_GPU"
```

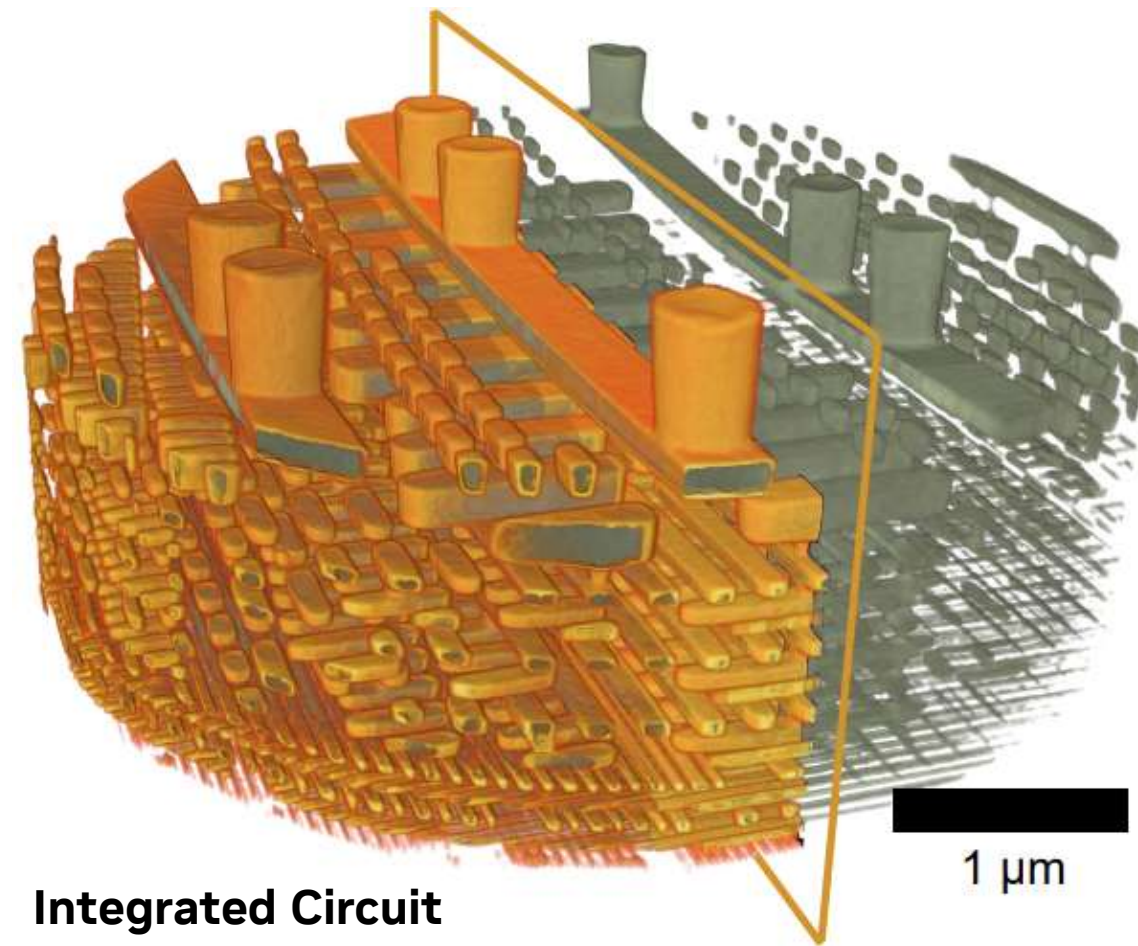
The background of the slide features a series of overlapping, curved, light green bands that create a sense of depth and movement, transitioning from a pale green on the left to a darker green on the right. A solid green vertical bar is positioned on the far left edge.

Materials Science

Nanoimaging

Exploring the Invisible: Unveiling Structures at the Nanoscale Through Advanced Imaging Techniques

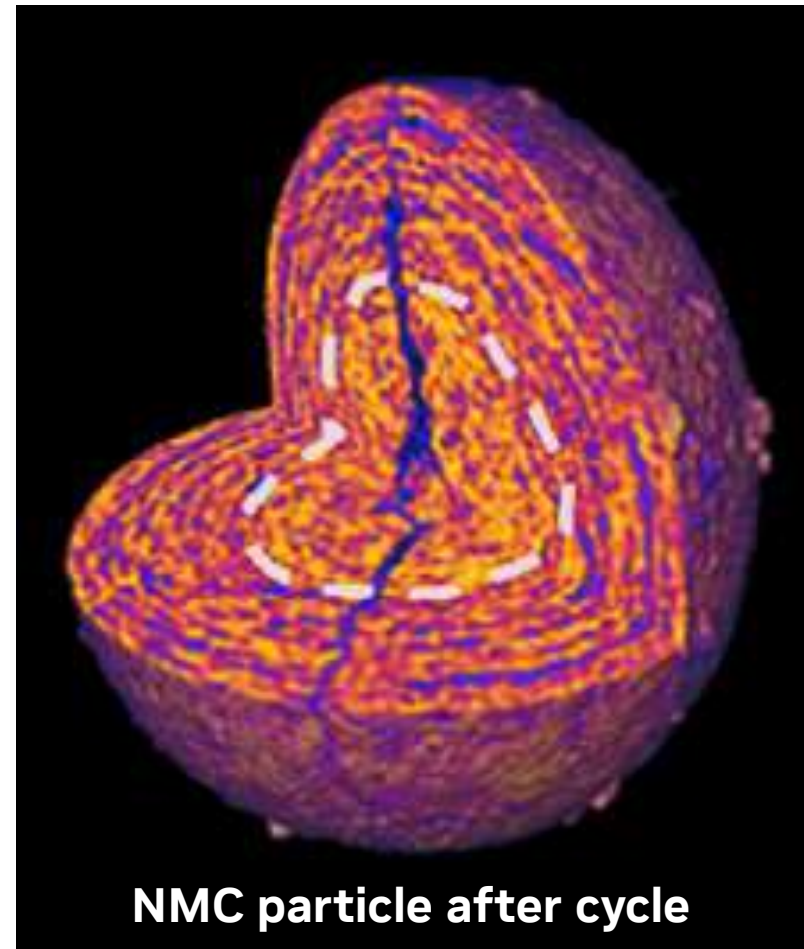
Electronics



Integrated Circuit

Aidukas, T. et al, Nature **632**, 81–88 (2024).

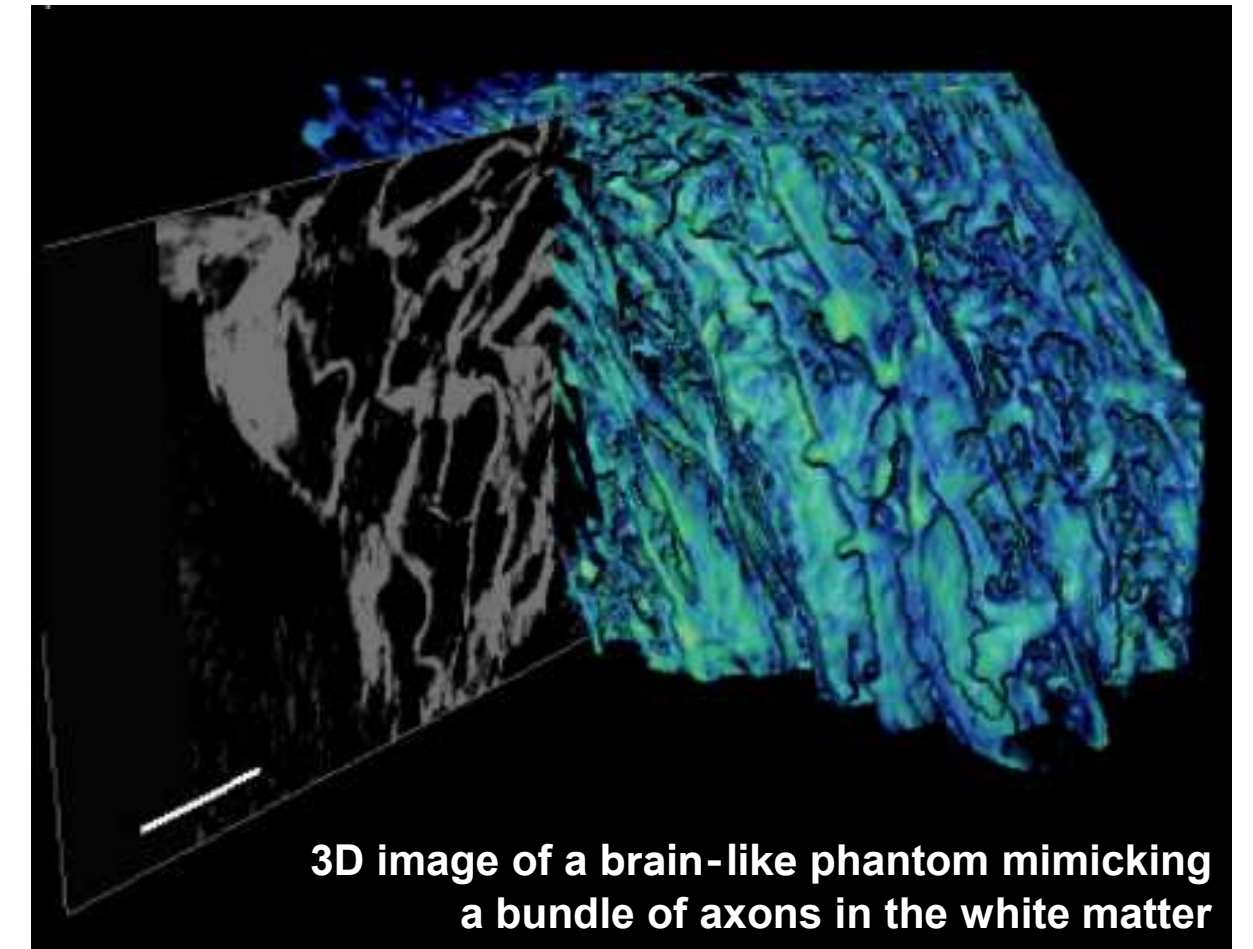
Batteries



NMC particle after cycle

(Tsai et al., iScience, 2019)

Life Sciences



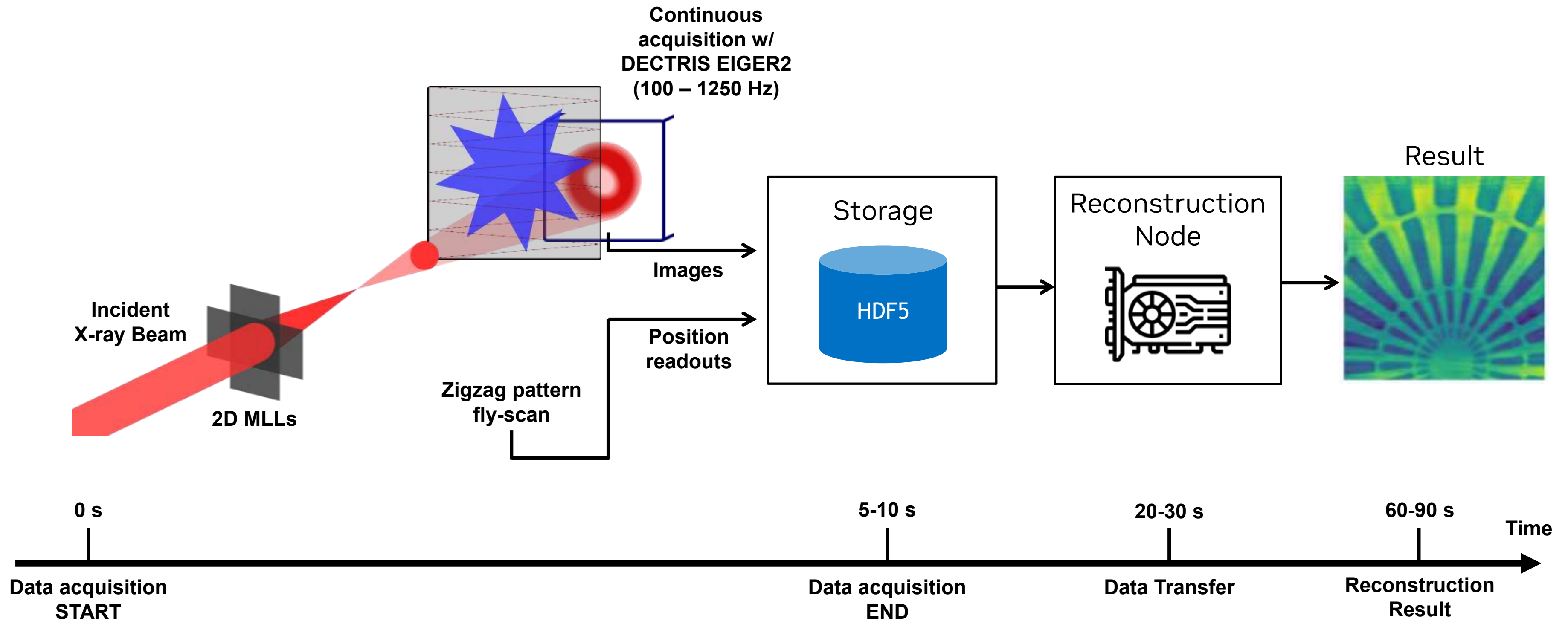
3D image of a brain-like phantom mimicking a bundle of axons in the white matter

Eur. Phys. J. Plus **139**, 434 (2024)

Nanoimaging seeks to reveal structure-function relationship in materials through high-resolution visualization of nm-μm features

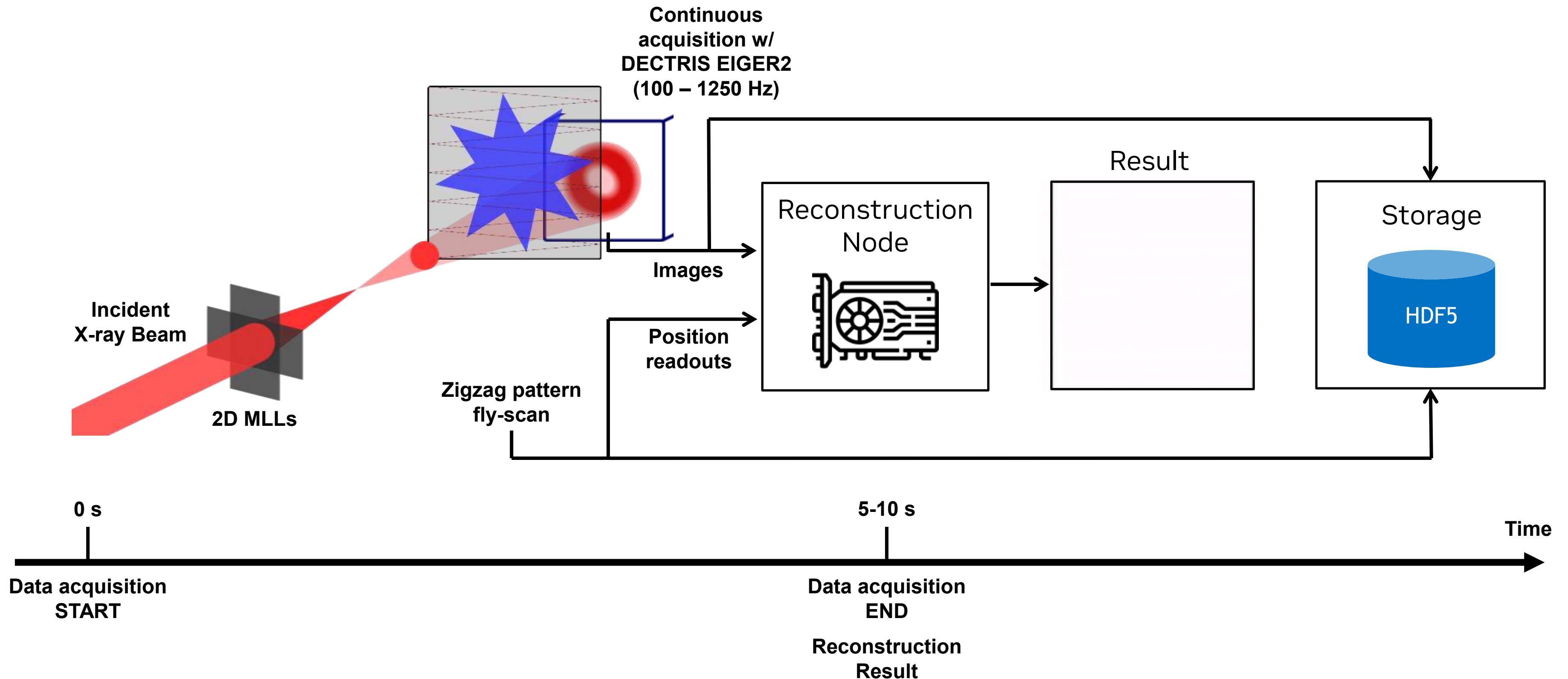
Ptychography at HXN before Holoscan

The Traditional Approach to Experiments



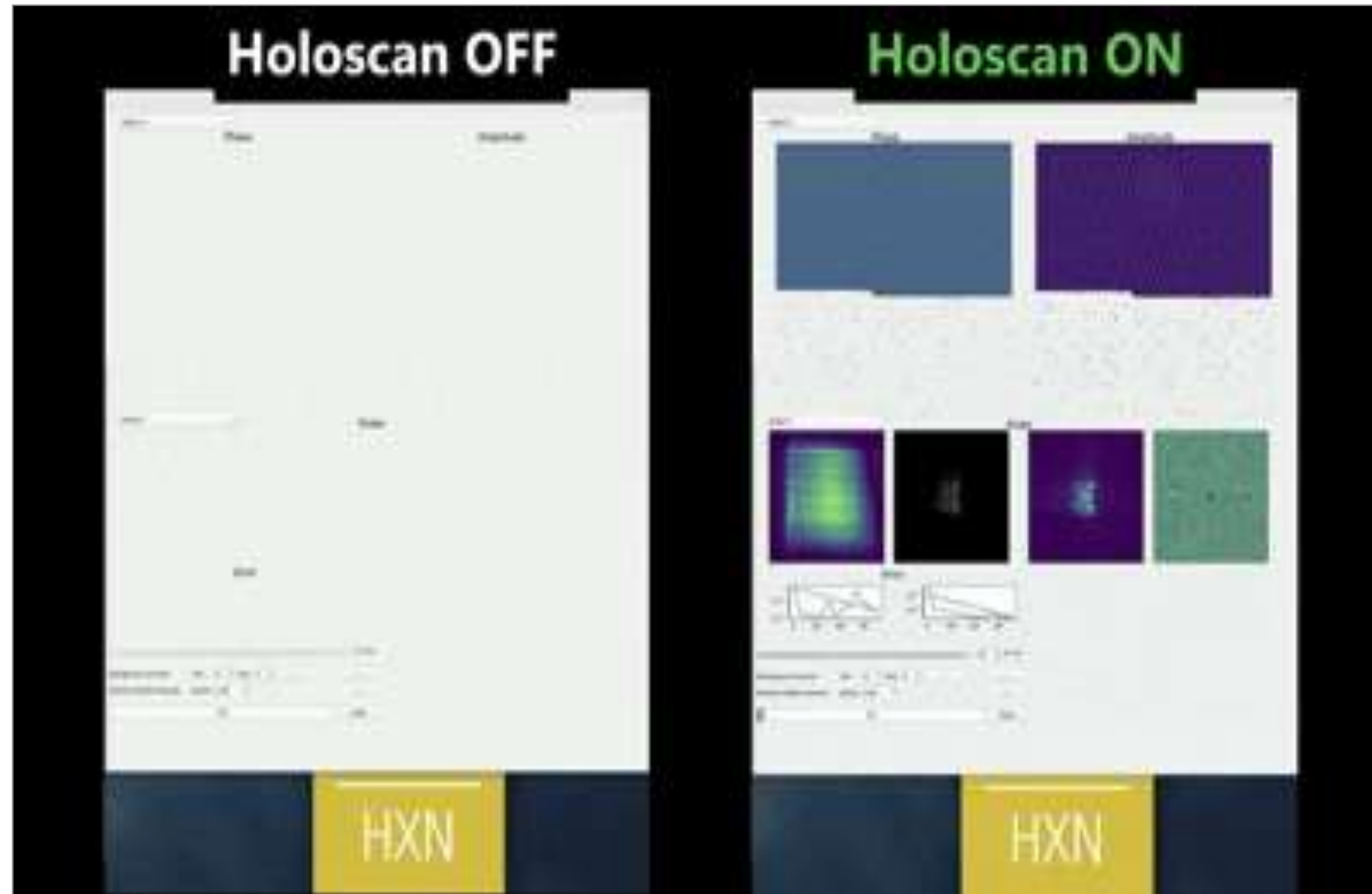
Ptychography at HXN with Holoscan

Real Time Analysis Feedback



Real Time Nanoscale Imaging with NVIDIA Holoscan

Brookhaven National Laboratory NSLS-II, DECTRIS



Demonstration of Offline versus Online Ptychographic Imaging

Self-assembly from nanoparticles, potentially useful as optical and mechanical metamaterials

For visualization purposes, both offline and online videos have been sped up by 2x

- **Current beamline scan takes about 8 seconds, with reconstruction taking 1-2 minutes**
 - Requires full dataset from x-ray detector and disk storage prior to reconstruction
- **Impact of Edge HPC and Holoscan for real time imaging**
 - Real time reconstruction provides instantaneous user feedback on experiment results
 - Ability to handle higher data rates yields higher resolution experiments and improved science products
 - Laying foundation for AI-driven agents and autonomous experimentation, including AI based scan patterns, leading to the **discovery of new materials at unprecedented rates**

Joint Hackathon: DECTRIS, NVIDIA, PSI, Diamond Light Source

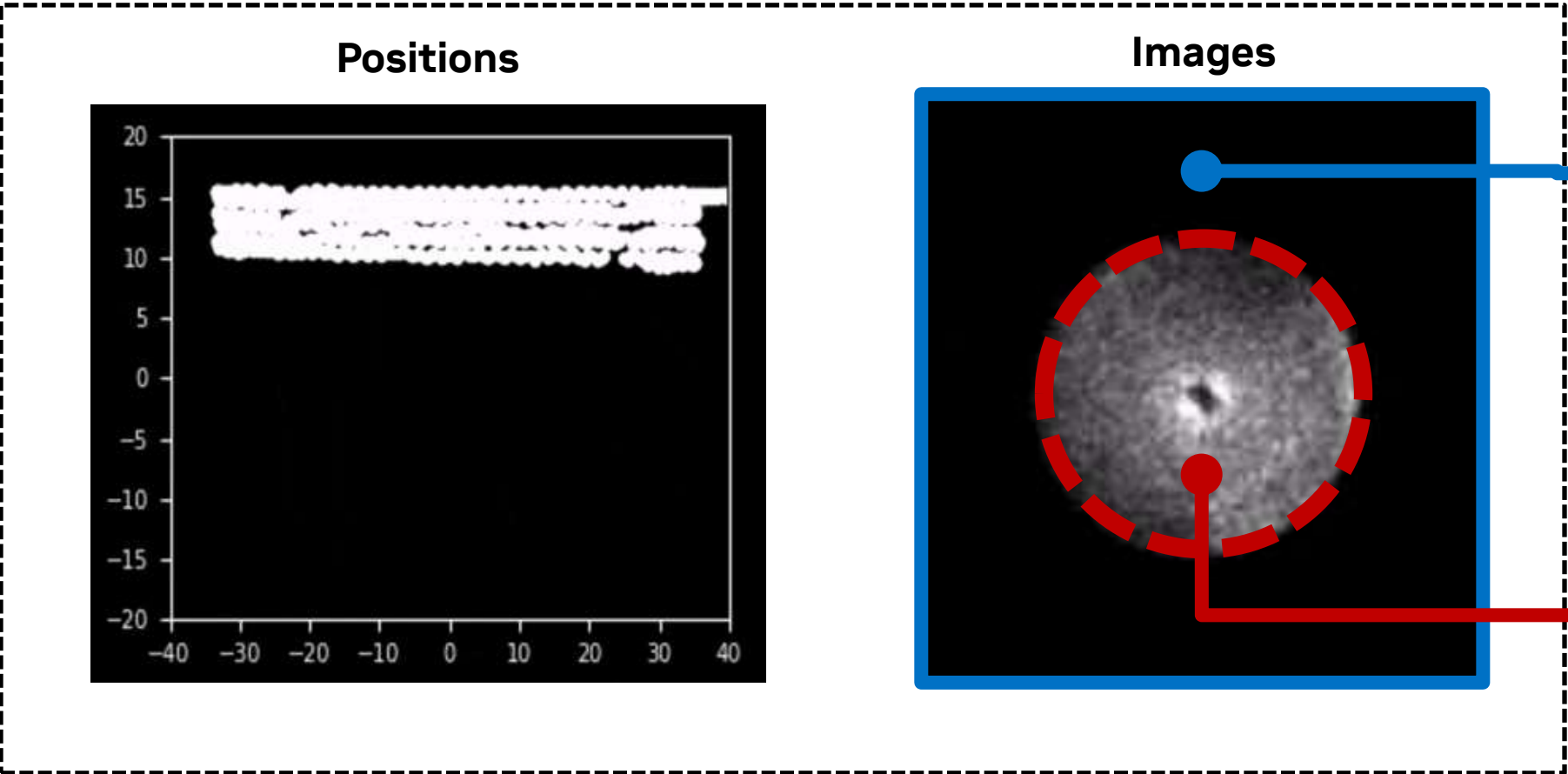
Taming the data streams from the next generation cameras

- Next generation cameras further speed up the data acquisition
- Hackathon was focused on enabling real-time feedback from data stream by analyzing images as in STXM experiment

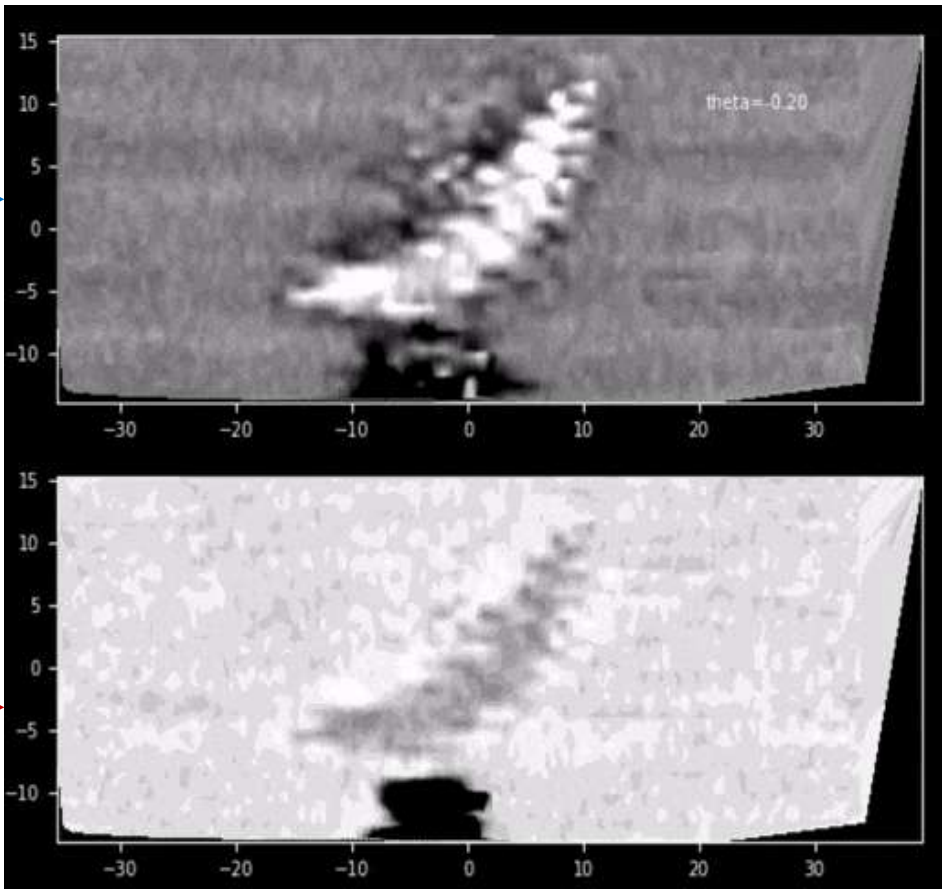
DECTRIS
SELUN Camera
Up to 120 000 FPS



Single Projection



Rotating Sample



HAADF

STXM

Deploying Holoscan with DECTRIS SELUN at I-13 beamline (DLS)

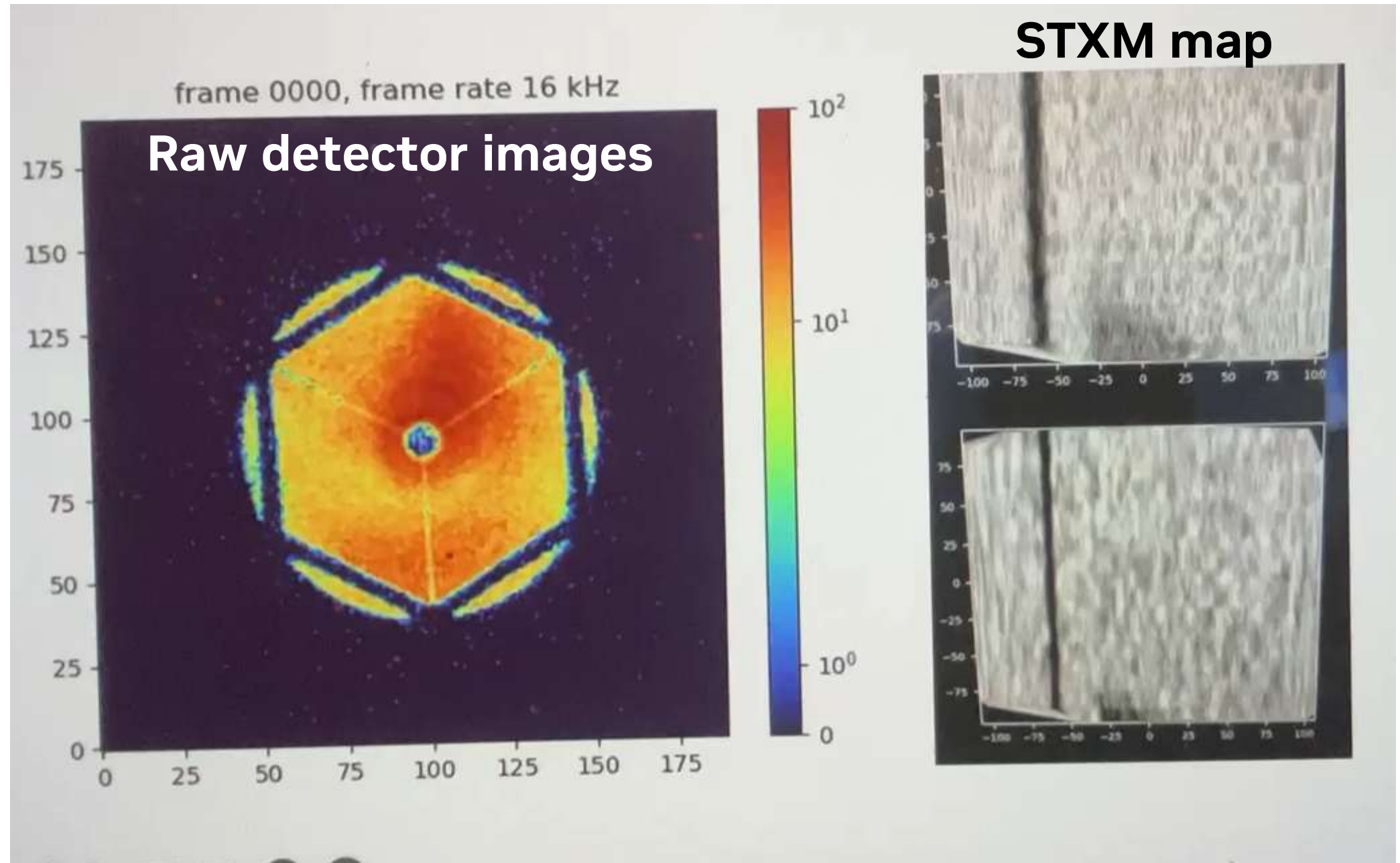
In Collaboration with DECTRIS

Results

- Real sample/real X-rays: up to 40 kHz - this is as fast as the beamline mechanical stages were able to go during testing
- Without sample motion: up to 60 kHz the pipeline kept up with data source; up to 80 kHz with enough buffering
- First integration with **DECTRIS CLOUD**
>> Poster 140 - Felix Bachmair

Next steps

- Real-time Ptychography with SELUN



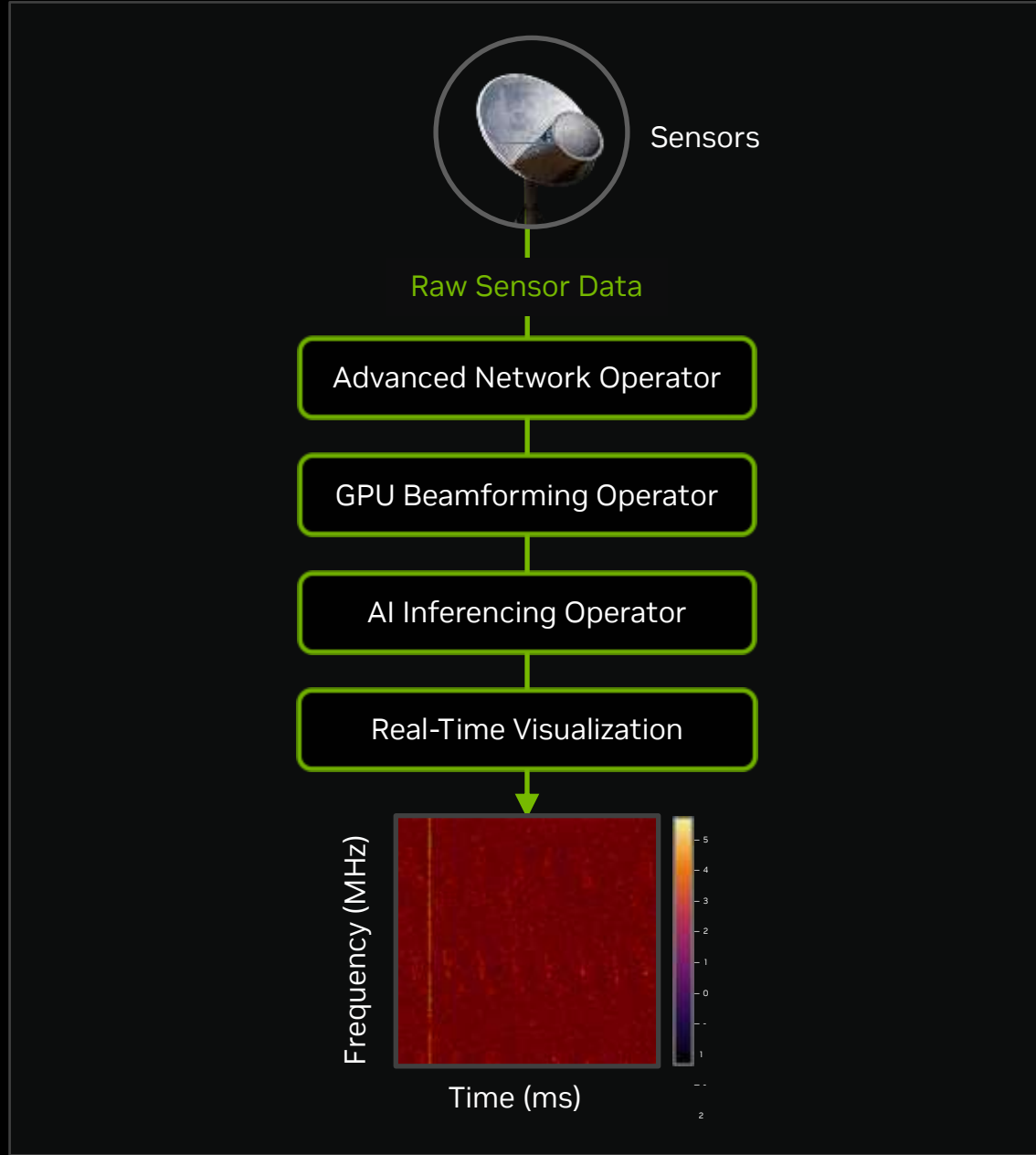
Radio Astronomy

First Real-Time Pure AI Detection of a Pulsar Using Raw Streaming Sensor Data

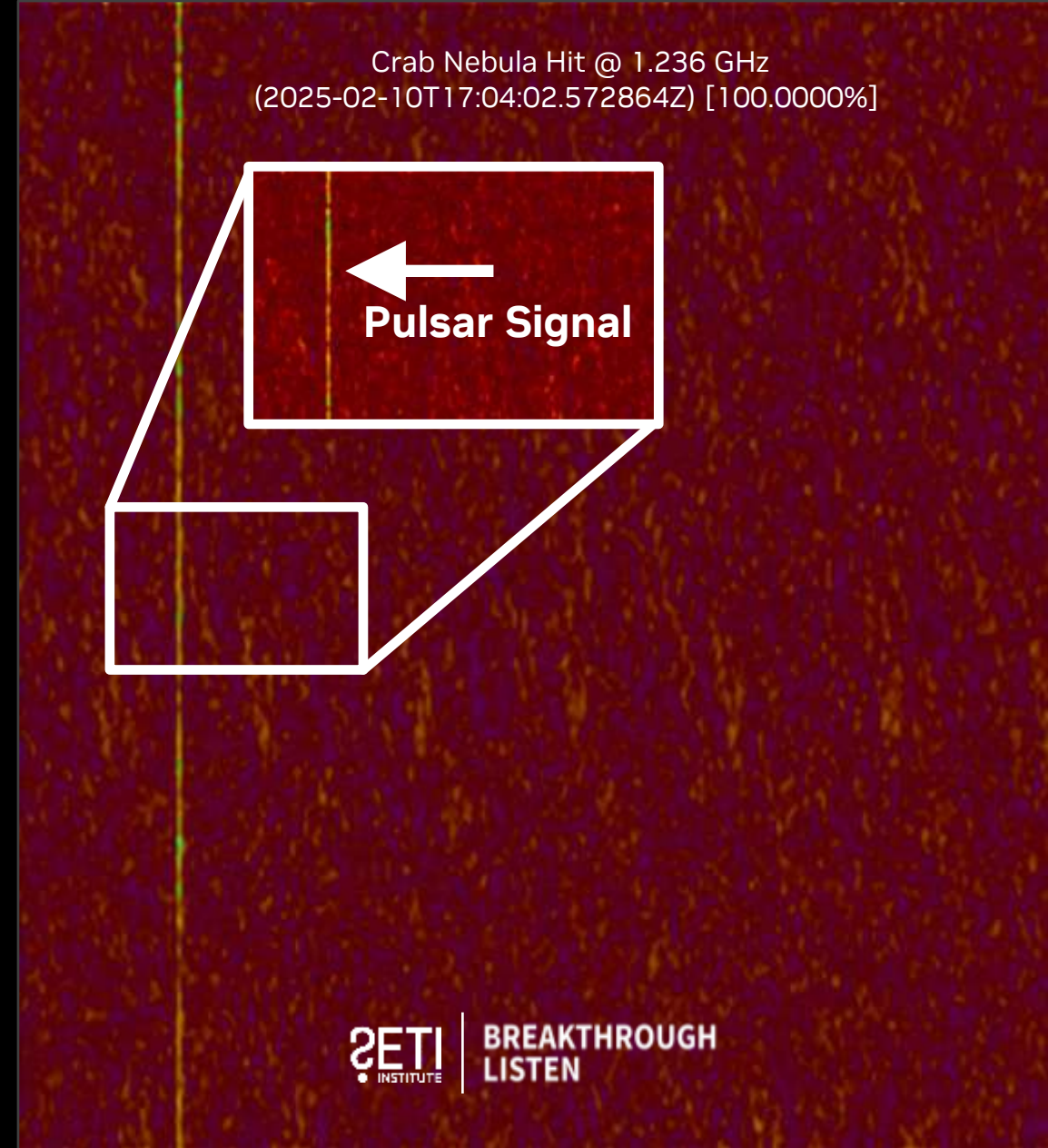
Holoscan Enables Real-Time AI-Powered Sensor Workloads at the Edge



Pulsar in the Crab Nebula



Holoscan From Beamformer to AI Model



Signal Detection Beyond Noise

600x Speedup | 160x Faster than Real Time | 10x Reduction in False Alarms

*Copyright NASA, SETI Institute



A New AI Digital Backend for Radio Astronomy



ASTRON – Westerbork Radio Observatory
All-Sky Monitor for Billion+ Star Search



NenuFar
French SKA Pathfinder – Fast Radio Burst Detection



GMRT
India SKA Pathfinder – Fast Radio Burst and Pulsar Detection

The background of the slide features a series of overlapping, curved, light green bands that create a sense of depth and movement, transitioning from a lighter shade on the left to a darker shade on the right. A solid green vertical bar is positioned on the far left edge.

GPU-Accelerated Signal Processing Libraries

cupyx.scipy.signal

GPU Accelerated Signal Processing with Python

formerly [cuSignal](#)

cupy.scipy.signal - Selected Algorithms

GPU-Accelerated SciPy Signal (Python)



Ambgfun
MVDR

	Convolution	Convolve/Correlate FFT Convolve Convolve/Correlate 2D
	Filtering and Filter Design	Resampling - Polyphase, Upfirdn, Resample Hilbert/Hilbert 2D Wiener Firwin, FIR Filter
	Waveform Generation	Chirp Square Gaussian Pulse
	Window Functions	Kaiser Blackman Hamming Hanning
	Spectral Analysis	Periodogram Welch Spectrogram
Phased Array		
Peak Finding		



[Full List of Supported Functions - CuPy Docs](#)

SciPy Signal – Polyphase Resampler

```
import numpy as np
from scipy import signal

start = 0
stop = 10
num_samps = int(1e8)
resample_up = 2
resample_down = 3

cx = np.linspace(start, stop, num_samps, endpoint=False)
cy = np.cos(-cx**2/6.0)

%%timeit
cf = signal.resample_poly(cy, resample_up, resample_down, window=('kaiser', 0.5))
```

2x Xeon E5-2600: 2.36 seconds

CuPy – Polyphase Resampler

```
import cupy as cp
import cupyx.scipy.signal as cusignal

start = 0
stop = 10
num_samps = int(1e8)
resample_up = 2
resample_down = 3

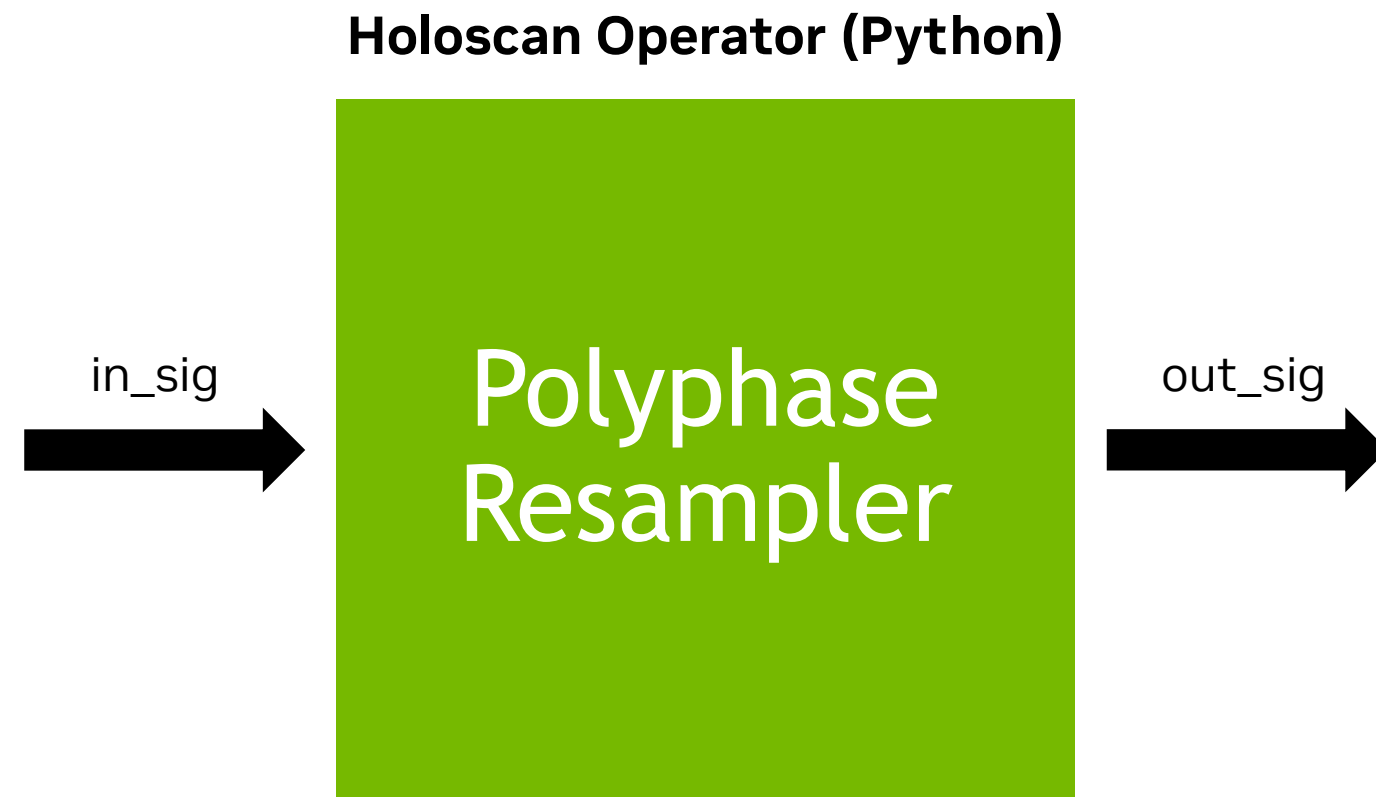
cx = cp.linspace(start, stop, num_samps, endpoint=False)
cy = cp.cos(-cx**2/6.0)

%%timeit
cf = cusignal.resample_poly(cy, resample_up, resample_down, window=('kaiser', 0.5))
```

NVIDIA A100: 4.69 milliseconds, **503x SciPy Signal (CPU)**

Seamless Integration with Holoscan

Connect Domain Specific Algorithms with Real Time Sensors



```
class ResampleOp(Operator):
```

```
    def __init__(self, *args, **kwargs):
```

```
        up = 2
```

```
        down = 3
```

```
        # Call base constructor class
```

```
        super().__init__(*args, **kwargs)
```

```
    def setup(self, spec: OperatorSpec):
```

```
        spec.input("in_sig")
```

```
        spec.output("out_sig")
```

```
    def compute(self, op_input, op_output, context):
```

```
        sig = op_input.receive("in_sig")
```

```
        resample_sig = cusignal.resample_poly(sig, up, down)
```

```
        op_output.emit(resample_sig, "out_sig")
```

MatX

**GPU/CPU-Accelerated C++
Numerical Computing**

MatX – C++17 Template Library for Numerical Computing

Design Overview

Features

Ease of Use:

Straightforward programming model with familiar interfaces (MATLAB/Python-like)

Wraps existing libraries like CUDA Math Libraries, NVPL, OpenBLAS, etc

Easily customizable

High Performance:

Prioritization of efficiently handling streaming data

Separates allocation and processing

Key Concepts

MatX leverages CUDA Managed Memory, freeing the developer from worrying about data locality

Compute operations are performed on arbitrary-rank **tensors** -- lightweight descriptors of data either on host or device.

Tensors are accepted in *all* MatX functions (like a NumPy ndarray)

Zero data movement view manipulations (clone, slice, permute) that can be chained together at compile-time

Supports many transforms: FFT, convolution, filtering, GEMM, pointwise operators, linear solvers, and more

Supports host and device execution with minimal code changes

MatX API Examples

Initialize a 2D tensor with data:

```
A = {{1, 2, 3}, {4, 5, 6}};
```

Add two tensor element-wise and scale:

```
(A = (A + B) / 5.0).run();
```

Perform a traditional GEMM:

```
(C = matmul(A, B)).run();
```

Perform an in-place scaled FFT:

```
(A = fft(A) / 256).run();
```

Batch sort a 4D tensor by rows:

```
(t4_sort = sort(t4)).run();
```

MatX/Python Comparison

FFT Based Resampler - No Windowing

Python

```
import numpy as np
from numpy import fft as fft

N = min(num_samp, num_samp_resamp)
nyq = N // 2 + 1

# Create an empty vector with num_samps elements
sig = np.empty(num_samp)

# Real to complex FFT, time to freq domain
fft_sig = fft.rfft(sig)

# Slice
slice_sig = fft_sig[0:nyq]

# Complex to real IFFT
resamp_sig = fft.irfft(slice_sig, num_samp_resamp)
```

5.36s (Xeon E5-2698v4 @ 2.20GHz)

MatX

```
uint32_t N = std::min(num_samp, num_samp_resamp);
uint32_t nyq = N / 2 + 1;

auto sigView = make_tensor<float, 1>({num_samp});
auto sigViewComplex = make_tensor<complex, 1>({num_samp/2+1});
auto resampView = make_tensor<float, 1>({num_samp_resamp});

// Real to Complex FFT, time to freq domain
(sigViewComplex = fft(sigView)).run();

// Slice to half spectrum based on num_samp_resamp
auto sliceView = slice(sigViewComplex, {0}, {nyq});

// Complex to Real IFFT, back to time domain
(resampView = ifft(sliceView)).run();
```

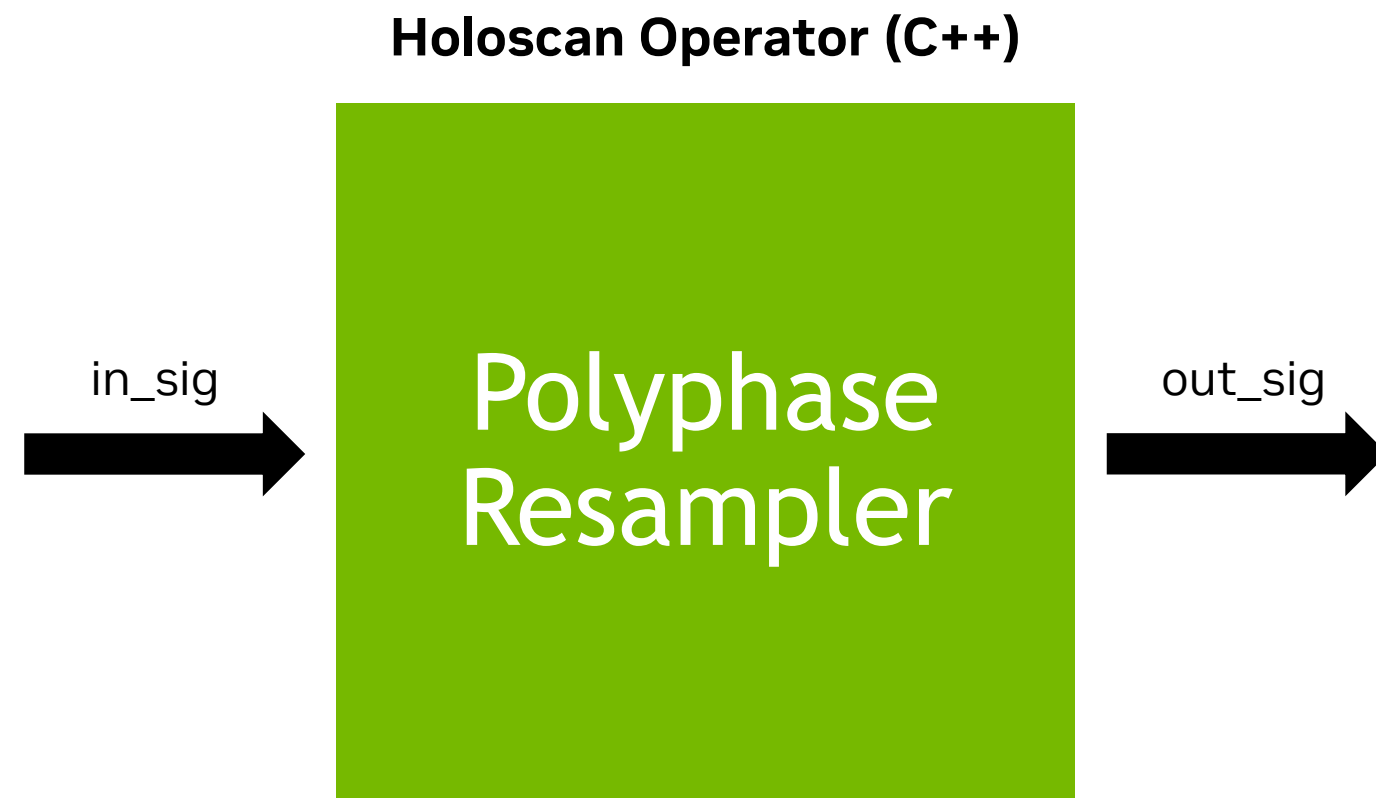
2.54ms (A100)



~2000x improvement!

Seamless Integration with Holoscan

Connect Domain Specific Algorithms with Real Time Sensors



```
class ResampleOp : public Operator {  
    void initialize() override {  
        N = 32768  
        # Call base constructor class  
        holoscan::Operator::initialize();  
    }  
    void setup(OperatorSpec& spec) override {  
        spec.input<matx::tensor_t<complex, 1>>("in_sig");  
        spec.output<matx::tensor_t<complex, 1>>("out_sig");  
    }  
    void compute(InputContext& op_input,  
                 OutputContext& op_output,  
                 ExecutionContext& ex) override {  
        auto sig = op_input.receive<matx::tensor_t<complex, 1>>("in_sig");  
        auto resample_sig = matx::resample_poly(sig, hanning<0>({N}), 3, 2);  
        op_output.emit(resample_sig, "out_sig");  
    }  
}
```

The background features a series of parallel, slightly curved lines in various shades of green, creating a sense of depth and movement. Overlapping, rounded rectangular shapes in different green tones are layered on top of these lines, adding a three-dimensional effect. The overall aesthetic is clean, modern, and tech-oriented.

Hardware Architectures for HPC and AI at the Edge

Hardware Architectures for HPC and AI at the Edge

Different Options for Different Sensor Requirements

Embedded Edge

Jetson Thor
40-130W
128GB
2070 TFLOPS (FP4)



Jetson Orin
7-65W
Up to 64GB
34 - 275 TOPS (INT8)



< 100 GbE I/O Thor
< 10 GbE I/O Orin

Industrial Edge

IGX Thor
130W - 430W
128GB + 96GB
2070 - 5581 TFLOPS (FP4)



IGX Orin
15-125W
64GB + 48GB
248 - 1705 TOPS (INT8)



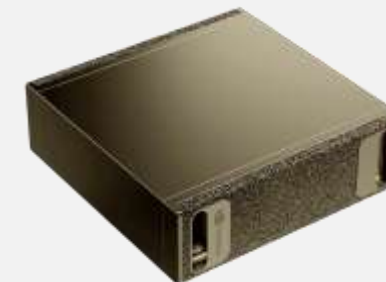
< 400 GbE I/O Thor
< 200 GbE I/O Orin

Enterprise Edge

Edge GPU
(RTX 6000 Pro/ L40S/L4)
40W-600W
Up to 96 GB
Up to 3700 TFLOPS (FP4)



DGX Spark/ Station
TBD
128 GB / 784 GB
1 PFLOPS (FP4) / 20 PFLOPS (FP4)



< 400 GbE
Scalable per NIC/GPU

Data Center / HPC Edge

GH 200
450W - 1000W
624GB
4 PF AI Perf



~1 Tbps I/O

Getting Started

Getting Started with Holoscan

Holoscan References



<https://github.com/nvidia-holoscan/holoscan-sdk>



```
docker pull nvcr.io/nvidia/clara-holoscan/holoscan:v3.4.0-dgpu
```



```
pip install holoscan  
conda install -c conda-forge holoscan
```



Debian Packages available on [NGC](#)



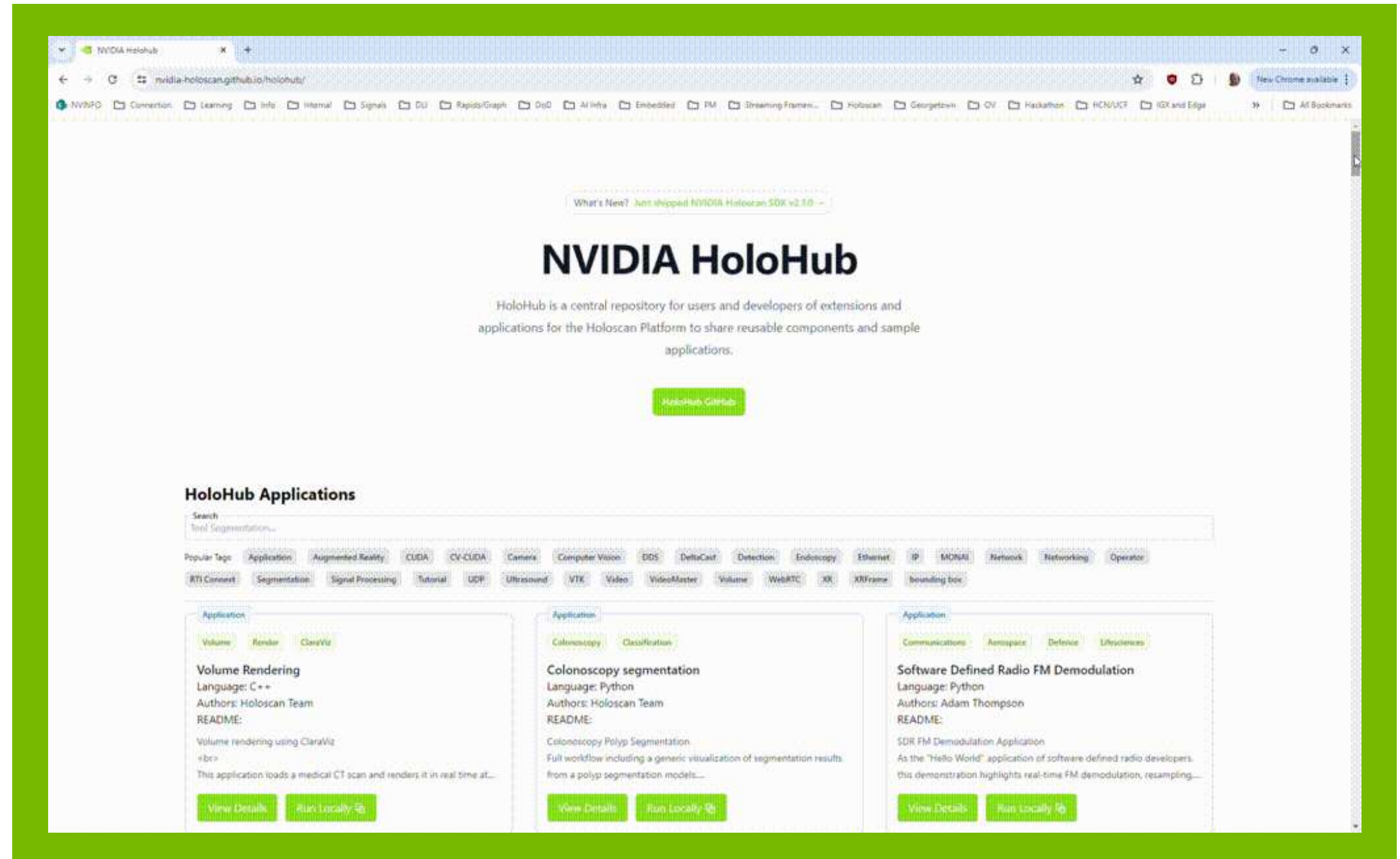
<https://docs.nvidia.com/clara-holoscan/sdk-user-guide/index.html>

Holoscan Reference Applications

Sample Holoscan Applications, Tutorials, and Helpful Operators



GitHub



IO: UDP Ethernet, Lidar, High Speed Cameras, SDR, SAR, 3D Volumes

AI: Segmentation, Tracking, AR/VR, LLMs

Tools/Tutorial: MATLAB, MONAI, Playground on AWS, Scaling Apps

