Future-Ready GEOINT: Hands-on Exploration of NVIDIA Holoscan's Sensor Processing Capabilities

Instructor:

Kevin Green and Chris Holland, Senior Solutions Architects for NVIDIA

Agenda

- Edge computing fundamentals & 4 pillars of real-time AI processing
- NVIDIA Holoscan Platform architecture and capabilities
- Real-world applications across healthcare, astronomy, and industries
- Hands-on workshop: Building custom Holoscan operators
  - Creating a simple data processing pipeline
  - Exploring the TAO PeopleNet Detection Model application

Edge Computing:
Intelligence at the Source

What is Edge Computing?

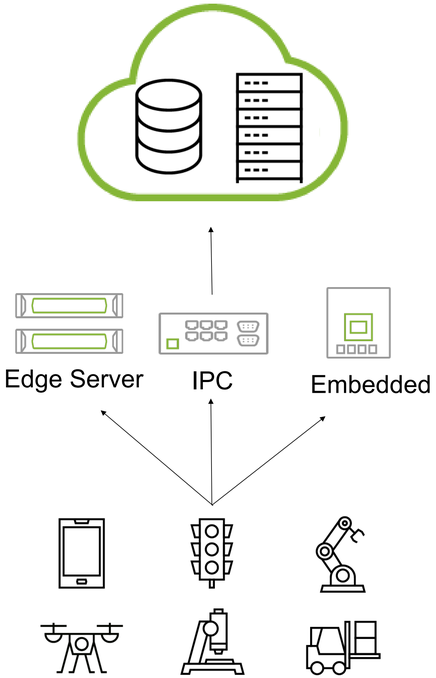
Low Latency, Reduced Bandwidth Requirement, Data Privacy and Improved Efficiency

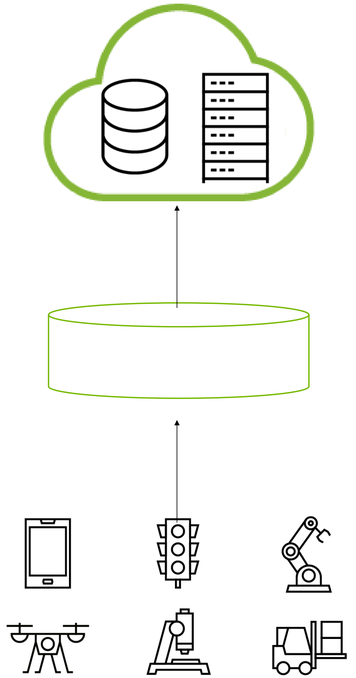
EDGE COMPUTING

Brings computation closer to the network edge where the data is gathered at source.

Average response time is milliseconds

Significantly reduced bandwidth





CLOUD COMPUTING

Location coverage is global because data centers are located around the world.

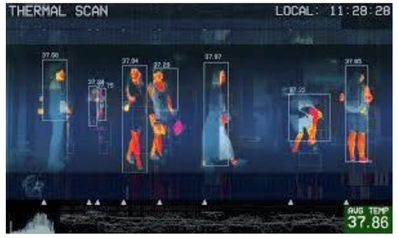
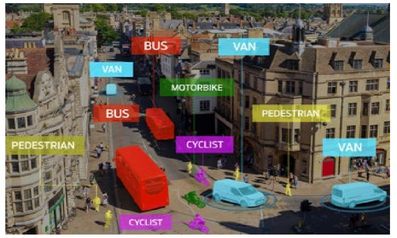
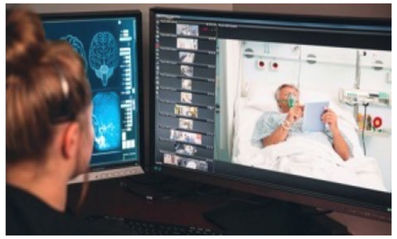
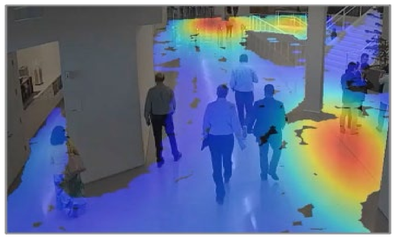
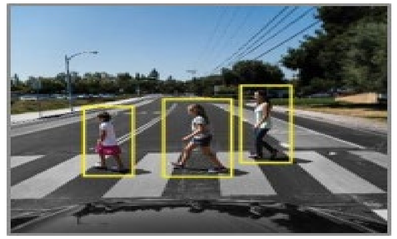
Average response time can still be milliseconds, but also minutes or days.

Requires a larger amount of bandwidth.

https://blogs.nvidia.com/blog/what-is-edge-computing/

Edge Computing Drives Intelligence For Every Industry

Industry-Specific Real-Time Applications

RETAIL	MANUFACTURING	HEALTHCARE	LOGISTICS	SAFETY
				
Loss Prevention	Quality Inspection	Body Temperature Screening	Efficient Fulfillment	Traffic Management
				
Store Analytics	Predictive Maintenance	Patient Monitoring	Safety Management	Contact Tracing
				
Stock Out and Replacement	Supply Chain Analytics	Surgery Analytics	Factory Floor Optimization	People Segmentation

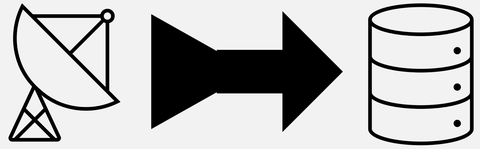
4 Pillars of Edge Computing

Benefits of Moving AI-Powered Computation Closer to the Sensor

REAL TIME INSIGHTS

Processing power is physically close to sensors collecting data, lowering time to insights

COST SAVINGS

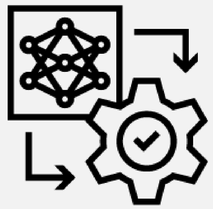


Filter and select prior to transmission and storage

INTELLIGENT EDGE AI

Collaborative edge to datacenter scaling and workflow

REQUIRES NEW TOOLS

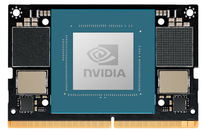


Ushering in a new age of software-defined and AI-enabled sensors. New tools are required to power new ideas

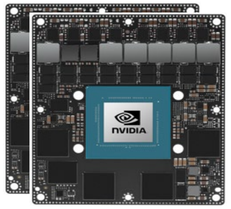
NVIDIA Holoscan The Scalable Edge Platform

Enabling Real Time, AI-Enabled Streaming Analytics at Any Scale

NVIDIA Holoscan



X86 + dGPU



Jetson Orin AGX

IGX Orin

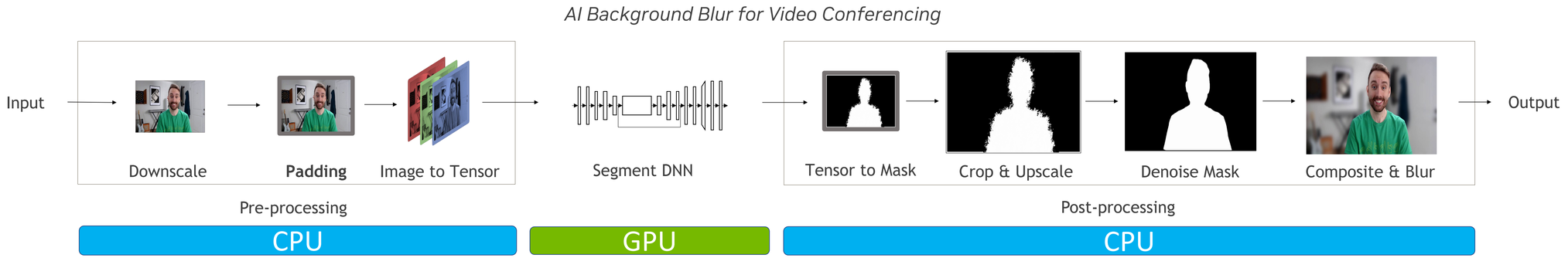
IGX Orin + dGPU

EGX/DGX

Embedded	Enterprise Edge	Cloud/Data Center

CPU Bottlenecks in AI/ML Pipelines

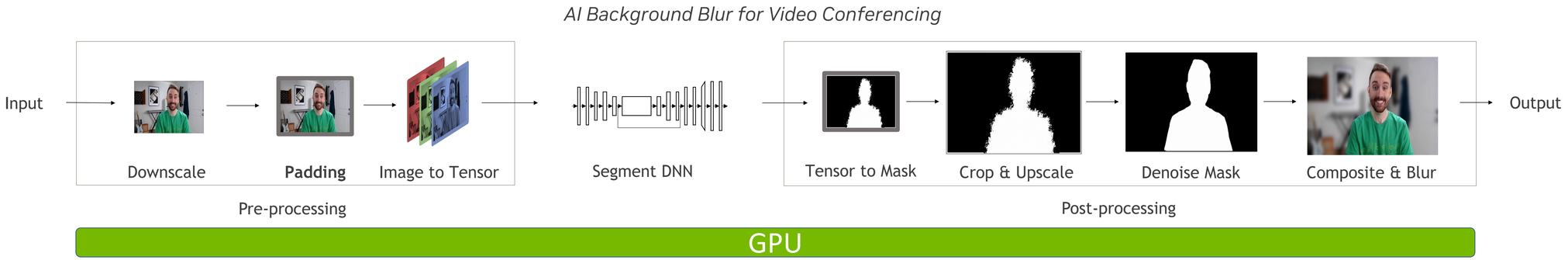
Conceptual Diagram: AI Imaging pipeline in the Coud/Data center Environments



1. **Data Loading**: Pre- and post-processing tasks account for roughly 90% of the workload.
2. **Processing Overhead**: The cost of these operations varies by task.
3. **Library Impact**: Python libraries (Torch, Keras) can inadvertently introduce CPU bottlenecks.

GPU Acceleration using Accelerator Frameworks

Accelerated end-to-end AI workflow on GPU



- **>10x throughput improvement** by moving entire pipeline on to the same single GPU hardware
- **Significant cloud costs savings** with accelerated software
- **No need to store intermediate storage products** reducing memory requirements

WIDE AREA LOCALIZATION OF DIGITIZED OBJECTS

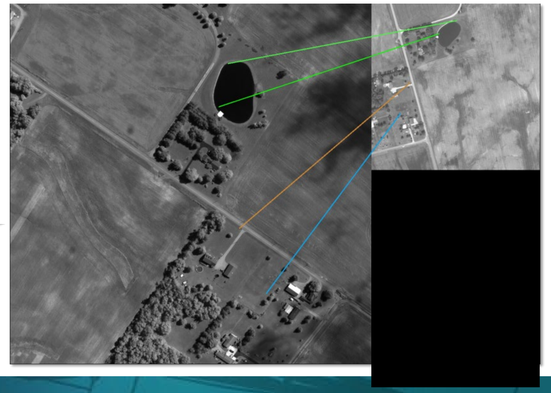
Georeferenced imagery is critical for national security, search and rescue, environmental monitoring.

Not all imagery contains geolocation metadata necessary for geo-registration pipelines. These images are manually geo-registered by GEOINT analysts. This process is cumbersome and extremely time intensive.

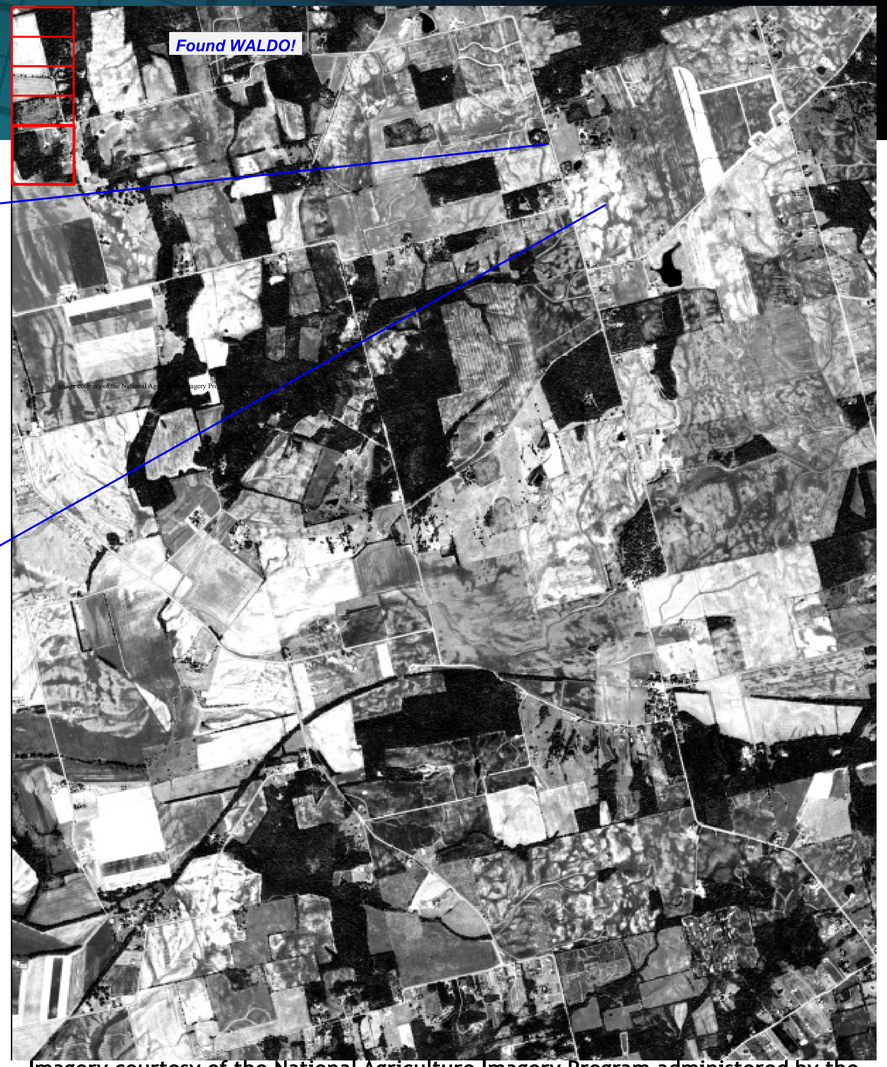
NGA required a computer vision application to rapidly geolocate un-geotagged aerial images within much larger geo-referenced aerial images representing huge regions of land mass.

NVIDIA and NGA Research collaborated on a solution to this continent scale image similarity search problem. Initially Implemented on CPU, processing took >24hrs. After GPU-acceleration, processing time was reduced to minutes – unlocking more GEOINT imagery for time sensitive missions.

Aerial Image without Geospatial Context



Search Large Continental Regions



Imagery courtesy of the National Agriculture Imagery Program administered by the USDA.

Introducing Holoscan:
NVIDIA’s Real-Time Streaming Sensor AI Platform

NVIDIA Holoscan vs. Alternative Approaches

Comparing Real-Time Sensor Processing Solutions

Traditional Approach

- CPU-bound processing bottlenecks
- Multiple independent components
- High latency between components
- Manual hardware optimization
- Complex integration of I/O

General AI Frameworks

- Optimized for batch processing
- Limited real-time guarantees
- Often requires custom integration
- Primarily designed for training
- Separate tools for deployment

NVIDIA Holoscan

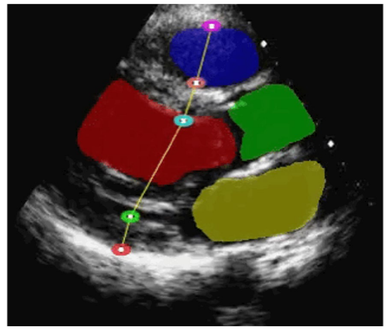
- Specialized for streaming data
- End-to-end pipeline optimization
- Deterministic low-latency
- Seamless GPU acceleration
- Unified sensor-to-insight stack

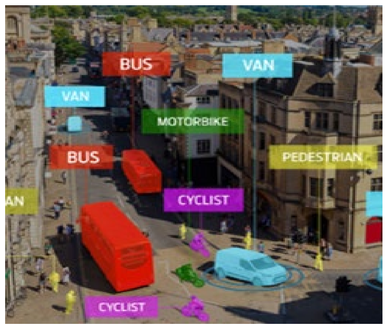
Key Differentiators:

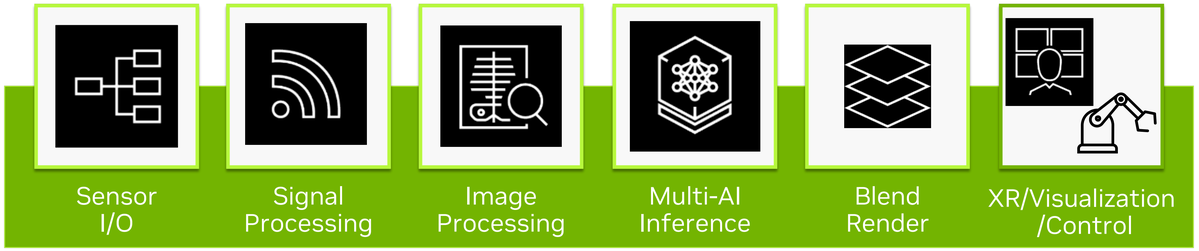
- Purpose-built for time-critical streaming sensor workflows
- Pre-optimized operators for common signal and image processing tasks
- Scale from embedded Jetson to datacenter seamlessly with the same code
- Domain-specific adaptability with 90+ reference applications

NVIDIA Holoscan

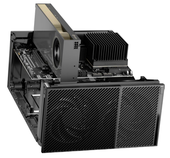
- **Universal Real-Time Sensing AI Platform**



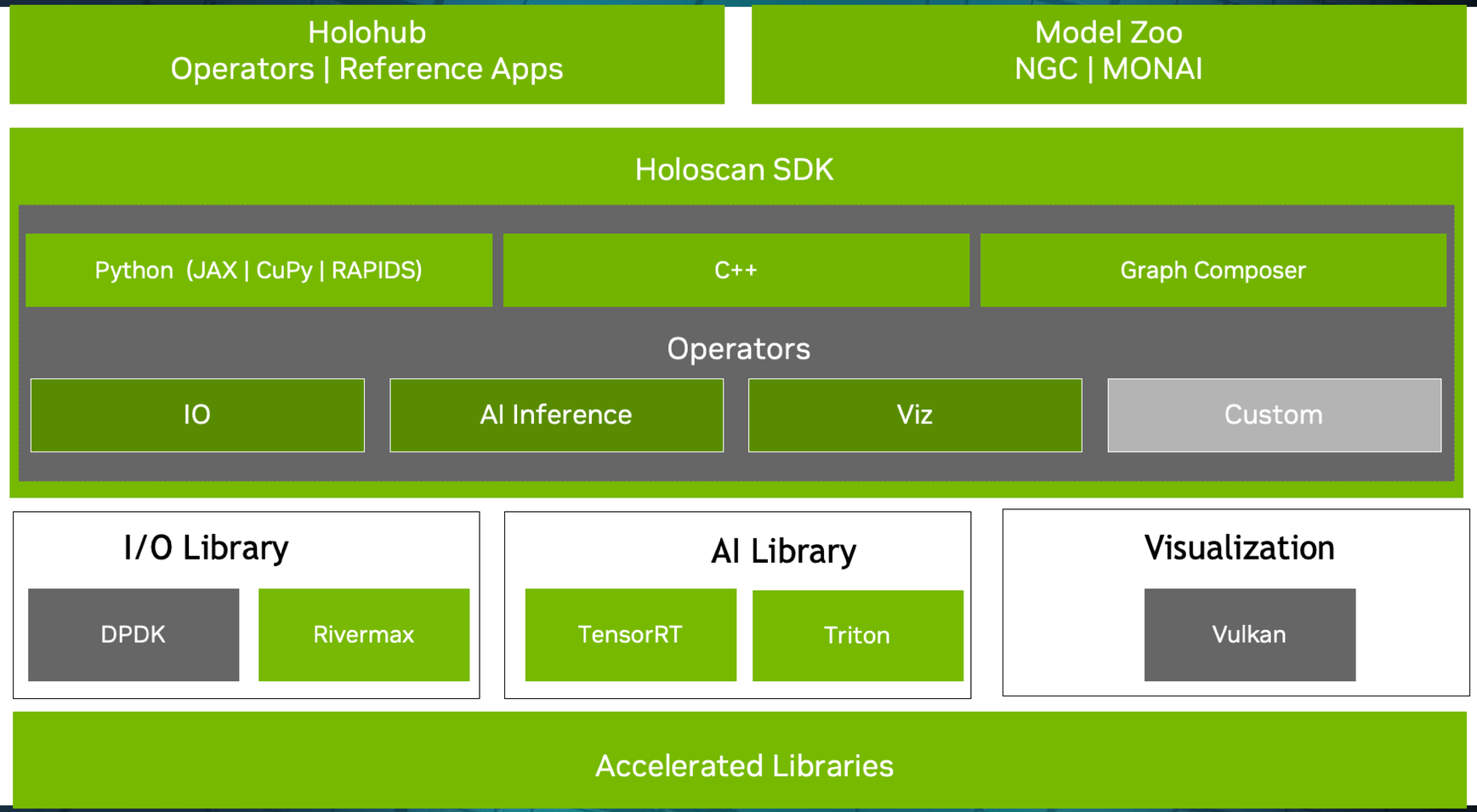

NVIDIA Jetson


NVIDIA IGX

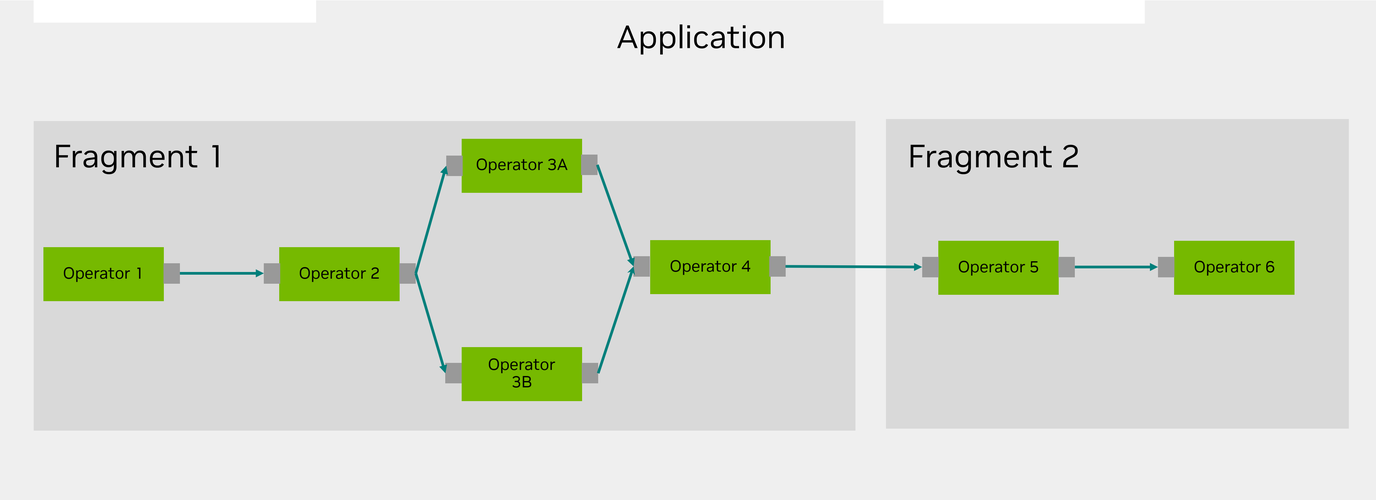

NVIDIA EGX

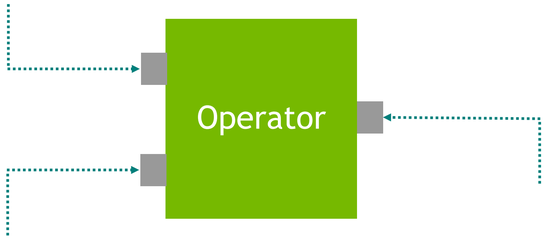
Holoscan Fundamentals

The software stack



Holoscan C++ and Python APIs





Application acquires and processes streaming data. It's a collection of fragments where each fragment can be allocated to execute on a physical node of a Holoscan cluster.

Fragment is a Directed Acyclic Graph (DAG) of operators. It can be assigned to a physical node of a Holoscan cluster during execution. The run-time execution manages communication across fragments. In a Fragment, Operators (Graph Nodes) are connected to each other by flows (Graph Edges).

Operator is the most basic unit of work. It receives streaming data at an input port, processes it, and publishes it to one of its output ports.

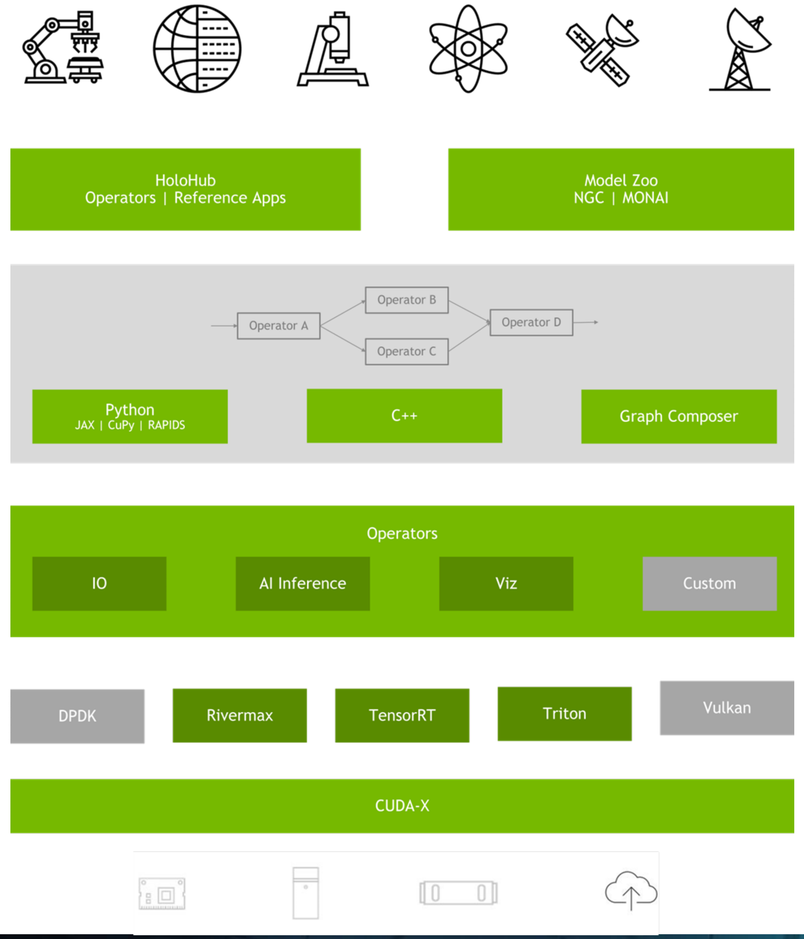
Port is an interaction point between two operators. Operators ingest data at Input ports and publish data at Output ports.

Condition is a predicate that can be evaluated at runtime to determine if an operator should execute.

Executor manages execution of a Fragment on a physical node.

NVIDIA Holoscan

Software Development Kit for Building AI-Enabled Sensor Processing Applications



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 DGX (x86 + H100)
- Apache 2 Licensed and Available on GitHub

Benefits

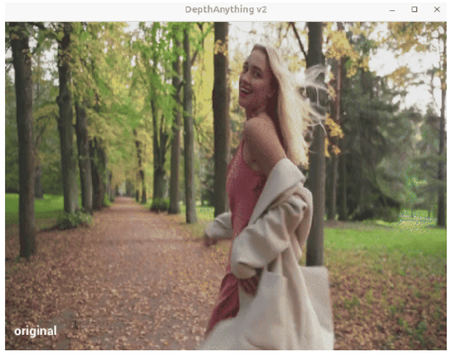
- Simplifies sensor I/O to GPU
- Simplifies the performant deployment of an AI model in a streaming pipeline
- Provides customizable, reusable, and flexible components to build and deploy GPU-accelerated algorithms
- Scale workloads with Holoscan's distributed computing features
- Deploy to the Cloud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

Holoscan in Action:

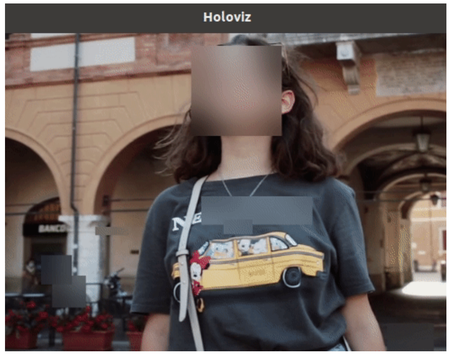
Real-World Applications Across Industries

Holoscan Reference Applications, Collection of 90+ Reference Applications

Jump start development with ready to use for testing and development application templates

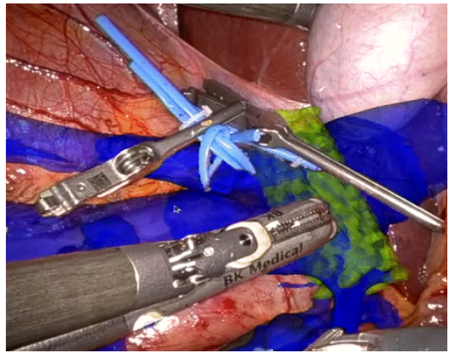


Depth Anything

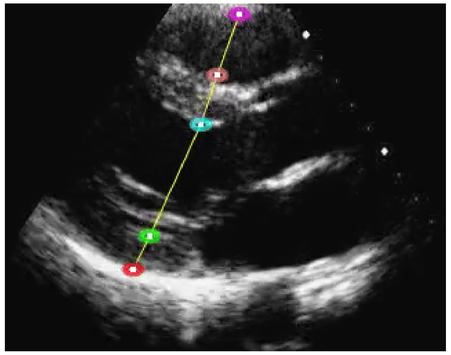


Real-Time Deidentification

Orthorectification Application

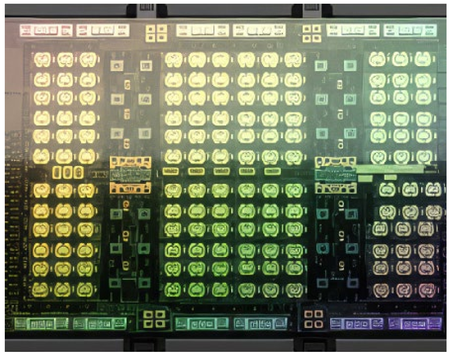


Real Time Surgical AI/AR



Multi-AI Ultrasound

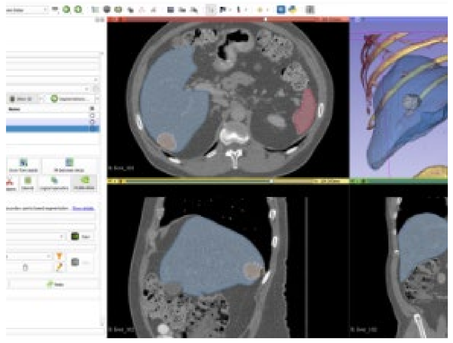
Synthetic Aperture Radar (SAR)



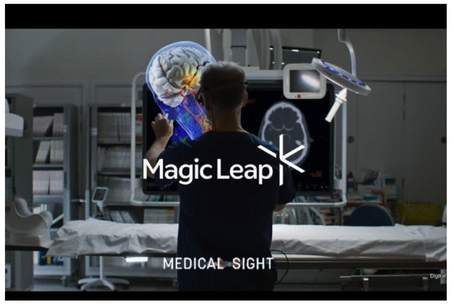
Sensor I/O: GPUDirect Data Ingestion

MathWorks

Bring Your Own MATLAB Algorithm



Real-Time Visualization With 3D Slicer



XR Volume Rendering

Dual-arm Surgical Robot Bedside Surgical Assistant

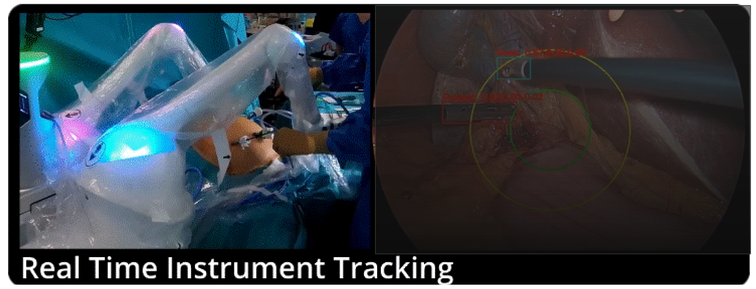
Next Wave of Surgical Robots: Copilots & Automation with Moon Surgical

Motion Control
(scope adjustment)

Live
Video

Preprocessing

ScoPilot



Subtask Automation

Prompt

ASR

Surgeon

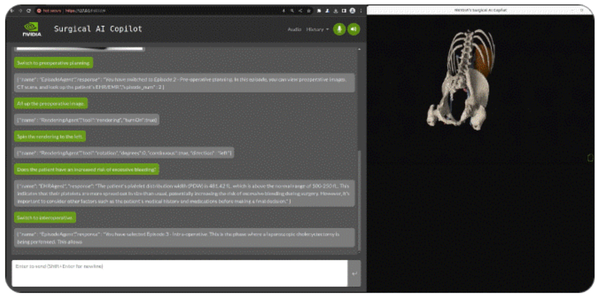
Response

TTS

Agent

NeMo Retriever

Embedding
Model



Agentic AI-powered Surgical Copilot

VILA VLM

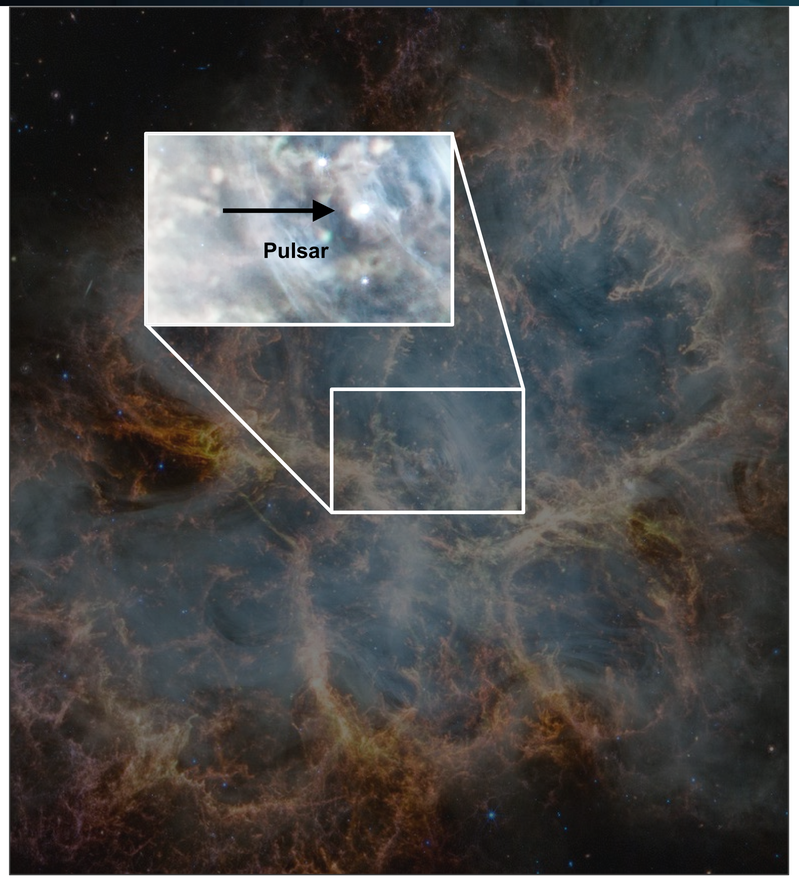
Vector
Database

EHR/PACS

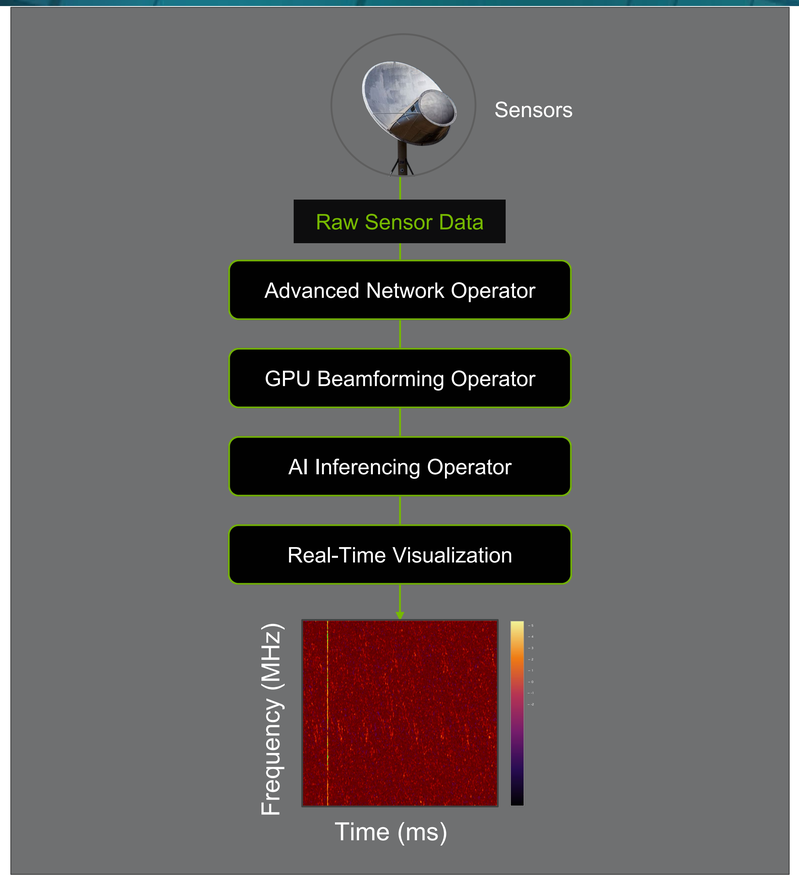
Holoscan – IGX

First Real-Time AI-Powered 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



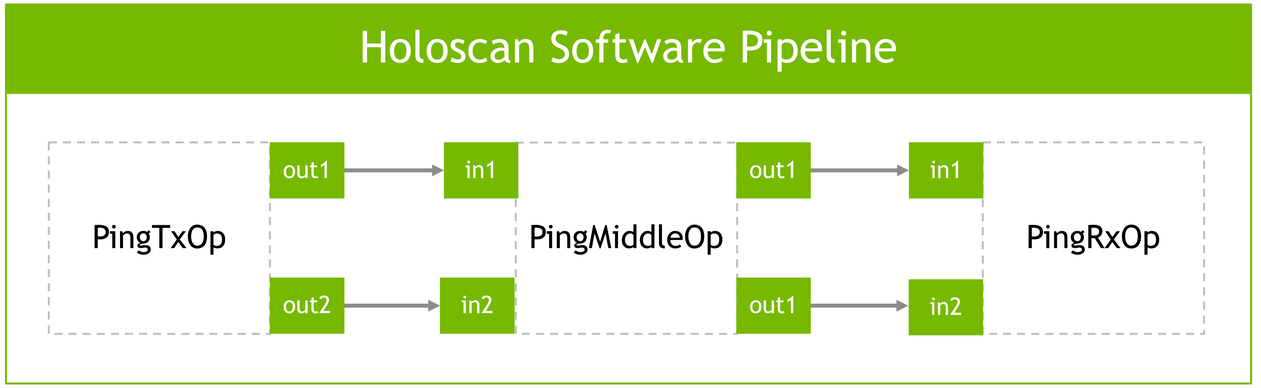
HOLOSCAN From Beamformer to AI Model

Signal Detection Beyond Noise

Building Your First Holoscan Application: From Theory to Practice

Our First Holoscan Application

Using Custom Operators to Create a Simple Application



Holoscan Workflow Example: Integer Stream Pipeline

This example illustrates Holoscan's core principles using a simple integer stream application with three custom operators:

- **PingTxOp**: Source operator emitting two integer streams via separate output ports
- **PingMiddleOp**: Transforms incoming data using mathematical operations
- **PingRxOp**: Sink operator printing processed results to the console

Holoscan Python API Deep Dive

Creating Custom Operators – Source / Transmitter

```python
class PingTxOp(Operator):
    """Simple transmitter operator.
    This operator has:
        outputs: "out1", "out2"
    On each tick, it transmits a `ValueData` object at each port. The
    transmitted values are even on port1 and odd on port2 and increment with
    each call to compute.
    """

    def __init__(self, *args, **kwargs):
        self.index = 0
        # Need to call the base class constructor last
        super().__init__(*args, **kwargs)

    def setup(self, spec: OperatorSpec):
        spec.output("out1")
        spec.output("out2")

    def compute(self, op_input, op_output, context):
        value1 = ValueData(self.index)
        self.index += 1
        op_output.emit(value1, "out1")

        value2 = ValueData(self.index)
        self.index += 1
        op_output.emit(value2, "out2")
```

Custom Operator extends Holoscan `Operator` class and includes 3 base functions: `__init__`, `setup`, and `compute`

Operator initialization/parameters

Name output ports

Emit message out of Operator via “out1”

Emit message out of Operator via “out2” port

Holoscan Python API Deep Dive

Creating Custom Operators – Middle Operator

```python
class PingMiddleOp(Operator):
    """Example of an operator modifying data.
    This operator has:
        inputs:  "in1", "in2"
        outputs: "out1", "out2"
    The data from each input is multiplied by a user-defined value.
    In this demo, the `multiplier` parameter value is read from a "ping.yaml"
    configuration file (near the bottom of this script), overriding the default
    defined in the setup() method below.
    """

    def __init__(self, *args, **kwargs):
        # If `self.multiplier` is set here (e.g., `self.multiplier = 4`), then
        # the default value by `param()` in `setup()` will be ignored.
        # (you can just call `spec.param("multiplier")` in `setup()` to use the
        # default value)
        #
        # self.multiplier = 4
        self.count = 1

        # Need to call the base class constructor last
        super().__init__(*args, **kwargs)

    def setup(self, spec: OperatorSpec):
        spec.input("in1")
        spec.input("in2")
        spec.output("out1")
        spec.output("out2")
        spec.param("multiplier", 2)

    def compute(self, op_input, op_output, context):
        value1 = op_input.receive("in1")
        value2 = op_input.receive("in2")
        print(f"Middle message received (count: {self.count})")
        self.count += 1

        print(f"Middle message value1: {value1.data}")
        print(f"Middle message value2: {value2.data}")

        # Multiply the values by the multiplier parameter
        value1.data *= self.multiplier
        value2.data *= self.multiplier

        op_output.emit(value1, "out1")
        op_output.emit(value2, "out2")
```

Custom Operator extends Holoscan `Operator` class and includes 3 base functions: `__init__`, `setup`, and `compute`

Operator initialization/parameters

Name input and output ports

Receive messages from input ports

Emit messages out of Operator via “out1” and “out2” port

HoloHub

A Repository for Hosting Sample Holoscan Applications and Operators

HoloHub seeks to turbo-charge Holoscan development by allowing engineering teams a “town-square” to easily contribute, share, and re-use new functionalities and demonstrate novel applications.

Repository: https://github.com/nvidia-holoscan/holohub

- Sample applications and operators are marked with a `metadata.json` file that describes the contribution, grades it for code quality, marks the supported compute platforms, lists dependencies, and provides domain tags (e.g. medical, industrial, aerospace)

- Prior to contributing to Holohub, please consult the contribution guidelines: https://github.com/nvidia-holoscan/holohub/blob/main/CONTRIBUTING.md

Transition to Jupyter Notebook: Hands-on Session

Learning objectives

- Create your own Holoscan operators with specific inputs and outputs that process sensor data in real-time
- Connect multiple operators to create complete data processing workflows
- Understand how Holoscan's fragment-based design enables efficient parallel processing and low-latency performance
- Use Holoscan's pre-built operators for sensor input, AI processing, and visualization to accelerate development
- Develop computer vision applications using NVIDIA's TAO PeopleNet model to perform person detection in video streams